A Fault-Tolerant Control Architecture for Unmanned Aerial Vehicles

A Thesis
Presented to
The Academic Faculty

by

Graham R. Drozeski

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
December 2005

A Fault-Tolerant Control Architecture for Unmanned Aerial Vehicles

Approved by:

Dr. George J. Vachtsevanos, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Bonnie Heck
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. David Taylor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Thomas Michaels
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. J. V. R. Prasad
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Eric N. Johnson
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: November 11, 2005

For my parents,

Thank you.

ACKNOWLEDGEMENTS

This research was supported by three Defense Advanced Research Projects Agency (DARPA) projects: Software Enabled Control (SEC), SEC Renegade (a follow-on to the SEC project funded to apply SEC technologies to a full-scale rotorcraft), and Heterogeneous Urban Reconnaissance, Surveillance, and Target Acquisition Team (HURT).

My deepest thanks are for Professor George J. Vachtsevanos, who advised me through the completion of this work. His critical input, guidance, and mentoring have been invaluable. I am also indebted to Professor Daniel Schrage, who advised my Master's work; Professor Eric Johnson, who directs the Georgia Tech Unmanned Aerial Vehicles Lab; Professor J.V.R. Prasad, who leads the SEC Renegade research effort; and the remaining members of my committee, Professor Bonnie Heck, Professor David Taylor, and Professor Thomas Michaels.

Several students and co-workers were also essential to the completion of this research: Suresh Kannan, Ben Ludington, Bhaskar Saha, Johan Reimann, Phillip Jones, Dr. N. Scott Clements, Alison Proctor, Dr. Luis Gutièrrez, Dr. Ilkay Yavrucuk, Dr. Andrew Gardener, Dr. Liang Tang, Dr. Seungkoo Lee, Suraj Unnikrishnan, Chen Chang, Nagabhushan Mahadevan, and Sharon Lawrence.

Lastly, my wife Michelle contributed to every page in ways she does not even know. I could not have done it without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

A	State transition matrix, continuous-time
A_f	State transition matrix for the fault-impaired system, continuous-time
$\tilde{A}$	Linear program constraint matrix
B	Control matrix, continuous-time
B_f	Control matrix for the fault-impaired system, continuous-time
C	Output matrix
$\bar{C}$	Expanded output matrix
F	State transition matrix, discrete-time
$\bar{F}$	Expanded state transition matrix, discrete-time
Fm	Indicator for the fault mode of the vehicle
G	Control matrix, discrete-time
$\bar{G}$	Expanded control matrix, discrete-time
K	Gain matrix
K_{act}	Actuator effective gain
K_f	Gain matrix for the fault-impaired system
M	Mission waypoint parameters
M_{com}	Commanded mission waypoint parameters
N	Length of optimization window
P, Q	Positive definite matrices, $A^T P + PA + Q = 0$
Pe	Aircraft performance function
R	Restructuring / reconfiguration indicator vector
U	Aircraft utility function
a_z, a_{rm_z}	Vertical vehicle and reference model accelerations
b	Actuator or neural network bias term
$\tilde{b}$	Linear program constraint vector

$\tilde{c}$	Linear program cost vector
d	Control equation bias / trim vector, discrete-time
$\bar{d}$	Expanded control equation bias / trim vector, discrete-time
e	Tracking or adaptation error vector
f	Linear program weight vector
g	Gravity
j_z	Vertical vehicle jerk
$\tilde{l}$	Linear program lower bound
p, q, r	Angular rates, body frame, radians/second
u, v, w	Velocity, body frame, feet/second
u	Control vector
$\bar{u}$	Expanded control vector, discrete-time
$\tilde{u}$	Linear program upper bound
w_{rm}	Vertical velocity reference model
x	State vector
$\bar{x}$	Expanded state vector, discrete-time
$\tilde{x}$	Linear program optimization variable
y	Output vector
$\bar{y}$	Expanded output vector
$\bar{y}_{com}$	Expanded command vector
z_{rm}	Vertical position reference model
Δ	Linear program flight path error bound vector
Ω	Main rotor angular rate, Revolutions per minute (RPM)
Ω_{com}	Main rotor angular rate command, Revolutions per minute (RPM)
Ω_{des}	Desired main rotor angular rate, Revolutions per minute (RPM)
δ_{aft}	Aft swashplate actuator input
δ_{coll}	Collective actuator input
δ_{lat}	Lateral cyclic actuator input
δ_{left}	Left swashplate actuator input

δ_{lon}	Longitudinal cyclic actuator input
$\delta_{min}, \delta_{max}$	Actuator position saturations
δ_{right}	Right swashplate actuator input
δ_t	Throttle actuator input
δ_{tr}	Tail rotor actuator input
ϵ	Model error
ζ	Damping coefficient
θ_{reg}	Regression estimate vector
ν	Pseudo-control
τ	Time constant of the main rotor RPM
ϕ_{reg}	Regressor vector
ϕ, θ, ψ	Euler angles, radians
ω_n	Natural frequency, radians/second

Acronyms

AMTC	Adaptive Mode Transition Control
API	Application Programmer Interface
CORBA	Common Object Broker Request Architecture
DARPA	Defense Advanced Research Projects Agency
DFT	Discrete Fourier Transform
FACT	Fault-Adaptive Control Technology
FDI	Fault Detection and Identification
FTC	Fault-Tolerant Control
FTR	Fourier Transform Regression
GTMax	Georgia Tech Yamaha RMax
HURT	Heterogeneous Urban Reconnaissance, Surveillance, and Target Acquisition Team
IFCS	Intelligent Flight Control System
IMF	Implicit Model Following
MMST	Multiple Models Switching Tuning
MPC	Model Predictive Control
MSLS	Modified Sequential Least Squares
NASA	National Aeronautics and Space Administration
PCH	Pseudo-Control Hedging
PID	Proportional Integral Derivative
RESTORE	Reconfigurable Control for a Tailless Fighter Aircraft
SEC	Software Enabled Control
SOM	Self Organizing Maps
SMC	Sliding Mode Control
TIFS	Total In-Flight Simulator
UA	Unmanned Aircraft
UAV	Unmanned Aerial Vehicle

SUMMARY

The success of unmanned aerial vehicles in recent military actions guarantees that their role in future operations will continue to expand. This expansion will routinely place unmanned vehicles in high threat environments and in close proximity to humans. Operation in both regimes requires an improvement in the current state of unmanned aerial vehicle reliability and fault-tolerance. The Office of the Secretary of Defense acknowledges this shortcoming in the *UAV Roadmap 2005-2030* by stating that, "Improving UA [unmanned aircraft] reliability is the single most immediate and long-reaching need to ensure their success" [2].

Research has presented several approaches to achieve varying degrees of fault-tolerance in unmanned aircraft. Approaches in reconfigurable flight control are generally divided into two categories: those which incorporate multiple non-adaptive controllers and switch between them based on the output of a fault detection and identification element and those that employ a single adaptive controller capable of compensating for a variety of fault modes. A limited number of fault-tolerant controllers combine these approaches thereby creating a continuously adaptive control framework that profits from the fault detection and identification process. Regardless of the approach for reconfigurable flight control, certain fault modes dictate system restructuring in order to prevent a catastrophic failure. System restructuring enables active control of actuation strategies not employed by the nominal system to recover controllability of the aircraft. Active control of a helicopter's main rotor RPM exemplifies this sort of restructuring. After system restructuring, continued operation of the aircraft in a degraded mode requires the generation of flight paths that adhere to an altered flight envelope. Reconfigurable methods for flight path planning are less prevalent in the literature. A comprehensive fault-tolerant control architecture for unmanned aerial vehicles should include active system restructuring and a method to re-shape the desired flight path of the aircraft.

The control architecture developed in this research employs a multi-tiered hierarchy to allow unmanned aircraft to generate and track safe flight paths despite the occurrence of potentially catastrophic faults. The hierarchical architecture increases the level of autonomy of the system by integrating five functionalities with the baseline system: fault detection and identification, active system restructuring, reconfigurable flight control, reconfigurable path planning, and mission adaptation. Fault detection and identification algorithms continually monitor aircraft performance and issue fault declarations. When the severity of a fault exceeds the capability of the baseline flight controller, the system actively restructures and selects one of multiple reconfigurable flight controllers. System restructuring expands the controllability of the aircraft using unconventional control strategies not exploited by the baseline controller. Each of the reconfigurable flight controllers and the baseline controller employ a proven adaptive neural network control strategy. The result is a suite of adaptive controllers each capable of accommodating an entire range of faults. A reconfigurable path planner employs an adaptive model of the vehicle to re-shape the desired flight path. Generation of the revised flight path is posed as a linear program constrained by the response of the degraded system. Finally, a mission adaptation component estimates limitations on the closed-loop performance of the aircraft and adjusts the aircraft mission accordingly.

Implementation of the fault-tolerant architecture on two separate unmanned helicopter airframes validates the utility of the hierarchical architecture. Active system restructuring includes active control of the main rotor RPM to accommodate faults in both collective and cyclic control actuators. Flight test and simulation results demonstrate the functionality of each component of the control architecture.

Major contributions of this research include:

- An integrated fault-tolerant architecture that incorporates fault detection and identification, active system restructuring, reconfigurable flight controllers, reconfigurable path planning, and mission adaptation to optimize the usefulness of the aircraft.
- An active system restructuring component that maximizes vehicle performance in the presence of a fault without degrading the performance of the nominal system.

- A suite of adaptive reconfigurable flight controllers with guaranteed stability properties that employ active control to augment the controllability of the degraded system.
- A reconfigurable path planning component that generates flight paths based on the capability of the degraded system.
- An online system identification process designed to enhance path planning and aid higher level decision making processes.
- A mission adaptation component that imposes constraints on the closed loop performance expectations of the aircraft.

CHAPTER 1

INTRODUCTION

Recent military actions around the world have underscored the potential utility of unmanned aerial vehicles (UAVs). Yet, most of their successes have occurred in a setting that allows a relatively large margin for error. UAVs have typically operated in flight profiles that maintain a large separation from obstacles and from other aircraft. As their roles continue to expand, UAVs will operate in progressively more challenging environments where this separation from flight hazards will be significantly reduced. Use of UAVs within the national airspace will require them to operate at a reliability that approaches or exceeds that of manned general aviation operations. General aviation averages one Class A mishap per 100,000 flight hours.[1] The Predator, which is currently the most reliable UAV in the U.S. fleet, averages 20 such mishaps per 100,000 flight hours [2]. Military usage in the urban warfare scenario calls for multiple UAVs operating simultaneously within the confines of an urban landscape. Operation in this environment requires incredible precision despite adverse flight conditions characterized by swirling winds and intermittent navigation signals. Furthermore, the close proximity to humans dramatically increases the risk associated with an in-flight mishap. The poor reliability of current unmanned vehicles presents a roadblock to their success in demanding new flight environments. The Office of the Secretary of Defense recognized this shortcoming in the *UAV Roadmap 2005-2030* identifying the development of "self-repairing", "smart" flight control systems as a crucial step in the overall advancement of UAV autonomy [2].

The challenge of developing an advanced fault-tolerant control system for UAVs can be approached on two levels. Modern UAVs are generally complicated, dynamic systems that possess several inter-connected components as well as an onboard computational capability. In this light, much research has been dedicated to the development of fault-tolerant

[1] Class A mishaps are those which result in a loss of aircraft, human life, or $1,000,000 in damage.

controls for complex systems. Dr. N. Scott Clements recently investigated the application of fault-tolerant control to a variety of complex dynamical systems [19]. His work included limited experiments conducted on a single UAV. While Clements' research does provide a foundation for addressing the UAV problem, its generality overlooks some of the details associated with unmanned flight. On another level, a UAV's control system should emulate a human pilot in its reaction to a fault condition. Of course, both industry and academia have devoted substantial research to improve the fault-tolerance of piloted aircraft. Reconfigurable flight control of manned and unmanned aerial vehicles has been an active area of research for several years. The research presented in sequel bridges these non-conflicting approaches to fault-tolerance on UAVs. The fault-tolerant control architecture developed by Clements' is modified to address the specifics of the UAV problem and to take advantage of recent accomplishments in reconfigurable flight control.

1.1 Discussion of Terms and Assumptions

Failures and Faults - A failure is defined as the "termination of the ability of an item to perform its specified function" [69]. In the context of this text, a failure is an event that prevents a UAV from sustaining controlled flight operations. Faults are physical defects that can precipitate a failure. Accordingly, the fault-tolerant control architecture developed in this research strives to prevent a vehicle level failure despite the presence of fault conditions. The architecture concerns itself specifically with faults that cause an unintended degradation in the performance of the system. Such faults are incipient failures, and the fault-tolerant control architecture acts to restore stability to the system. Table 1 copied from the *UAV Roadmap 2005-2030* attributes the majority of UAV failures to Power/Propulsion and Flight Control.

Restructuring and Reconfiguration - Fault-tolerant controllers employ both system restructuring and control reconfiguration to improve system performance in the presence of a fault mode. Restructuring implies changing the inter-connections within the system to achieve a new method of operation. Restructuring can occur in the system's hardware

Table 1. Summary of UAV Failure Mode Findings (Table H-2 from the *UAV Roadmap 2005-2030*).

		Power / Propulsion	Flight Control	Comm	Human / Ground	Misc
Aircraft	RQ-1A / Predator	23%	39%	11%	16%	11%
	MQ-1B / Predator	53%	23%	10%	2%	12%
	RQ-2A / Pioneer	29%	29%	19%	18%	5%
	RQ-2B / Pioneer	51%	15%	13%	19%	2%
	RQ-5A / Hunter*	38%	5%	31%	7%	19%
	RQ-7 / Shadow	38%	0%	0%	38%	24%

* The vast majority of all Hunter aborts 58% were due to weather.

or software. System restructuring typically dictates an associated control reconfiguration. Reconfiguration involves adaptation within the control laws of the system. This use of the terms coincides for the most part with Clements' although other sources present conflicting definitions [19, 29].

Active Control - Active control implies the use of unconventional control methods to improve the controllability of a system. Active systems "change their characteristics in real-time for better safety and higher efficiency" [66]. The control authorities employed in active control are generally beyond the intent of the original system designer. Achieving active control involves system restructuring, and the application of control efforts not employed by the nominal system.

Waypoints and Flight Paths - In the simplest form, waypoints are specified spatial and temporal locations for the vehicle. They command the vehicle to be in the proximity of a certain point at a certain time. Human UAV operators often form missions for UAVs as a sequence of waypoints, separated by several seconds up to minutes in time. Individual waypoints commonly have parameters, such as a target velocity, associated with them to facilitate path planning. Path planning algorithms translate waypoints into flight paths, which connect waypoints and form a continuous sequence of position commands for vehicle. Flight paths provide a command input for the low-level flight controllers at every time step.

Assumptions - The fault-tolerant control architecture for UAVs applies system restructuring to address a pre-determined set of detectable faults. The architecture can also

accommodate select undetectable faults that do not require system restructuring. The architecture also requires sufficient control authority in the fault-degraded system to recover the aircraft. Depending on the vehicle, several categories of faults may meet these criteria; however, this research addresses actuator malfunctions specifically. Actuator malfunctions affect the aircraft in a complex manner that stimulates every component of the fault-tolerant architecture. Most faults do not demand such a comprehensive modification to the control architecture. Additionally, a large variety of actuator malfunctions are easily simulated in flight without adding hardware to the airframe. On Table 1, actuator faults are classified under Flight Control. Flight Control also includes avionics, air data systems, control surfaces, onboard software, and navigation packages [1].

To combat malfunctions in the flight control actuators, the fault-tolerant architecture assumes the following actuator model:

$$\delta = max[min(K_{act}\delta_{com} + b, \delta_{max}), \delta_{min}] \tag{1}$$

where faults can affect the actuator effective gain $K_{act} : 0 \leq K_{act} \leq 1$, bias b, or saturation levels, δ_{min} and δ_{max}. Faults, such as floating actuators where the parameters vary constantly following the occurrence of the fault, are not considered. To accommodate these cases, additional hardware on the vehicle could be employed to immobilize the actuator creating a stuck actuator condition ($K_{act} = 0$).

1.2 Fault-Tolerant Control of Complex Systems

Large-scale systems such as manufacturing facilities, naval vessels, and aircraft generally include several integrated systems and sub-systems. Along with multiple components, these complex systems exhibit multiple modes of operation. Automated controllers oversee the operation of the system and direct changes in the system mode of operation. Both the environment and the health of the system can initiate undirected changes in system mode of operation. When the cause of a disturbance is system health, fault-tolerant control methods such as fault isolation, system restructuring, and control reconfiguration provide a means for stabilizing the system.

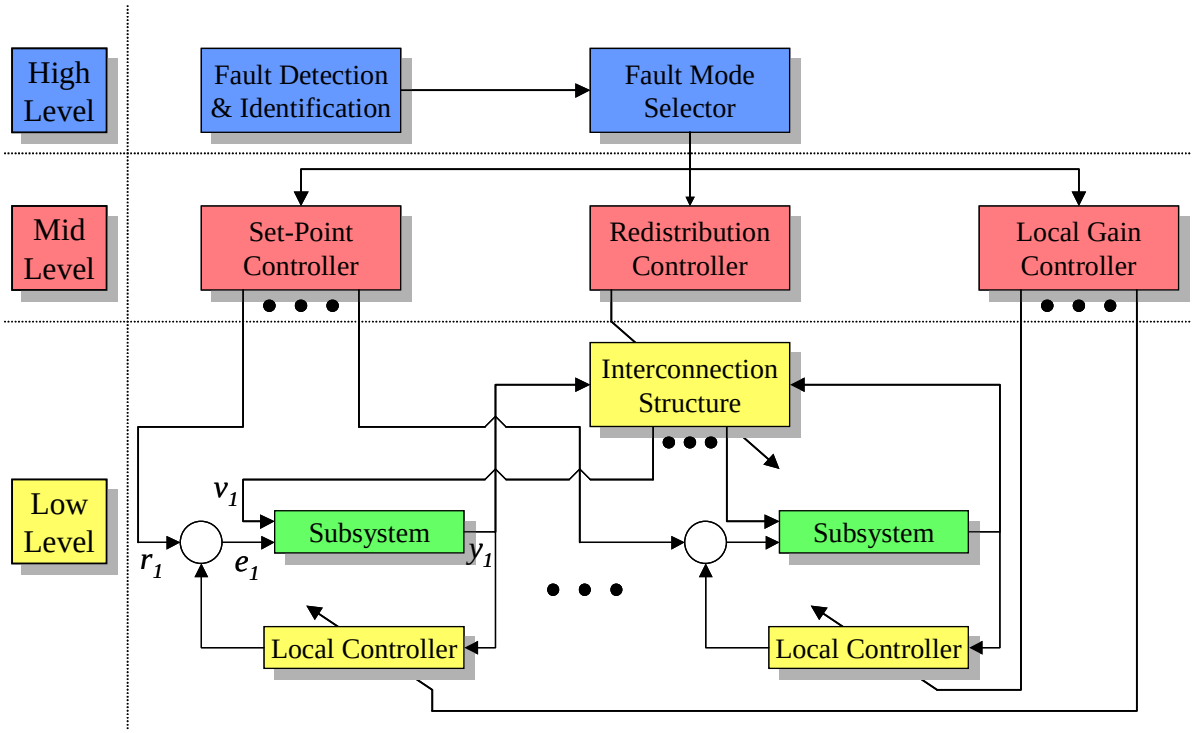

Figure 1. Clements' hierarchical fault-tolerant control architecture.

Fault-tolerant control of complex dynamical systems requires the integration of several functionalities and possibly several controllers in an adaptive control scheme. The complexity of the integration process is simplified by the use of a multi-layer hierarchical control scheme. Several researchers have implemented a very similar architecture [13, 19, 32]. Figure 1 depicts the fault-tolerant control architecture implemented by Clements [19]. Each level of the control hierarchy adds autonomy to the architecture. The scheme is readily expandable vertically and horizontally. In his architecture, fault detection and identification (FDI) as well as fault isolation occurs at the third tier of the hierarchy. Fault isolation includes restructuring to prevent propagation of the fault to healthy components. Following fault isolation, the middle tier of the hierarchy conducts three tasks to optimize the system's response to a fault. First, the redistribution controller restructures system inter-connections to optimize the use of control authority. "The goal of optimization [in the redistribution controller] is to maximize the routing of control authority between the subsystems." [19]. Then, based on the output of the redistribution controller, the set-point controller adjusts the set-points of the low-level controllers. Finally, the low-level gain controller adjusts the gains of the low-level controllers as necessary. Clements' architecture includes all the essential elements of a fault-tolerant control architecture at a level of generality that allows its application to a wide variety of systems. However, implementing the architecture on a specific system requires the development of algorithms to perform each of its separate functions. In the case of UAVs, for instance, this development is non-trivial.

1.3 Fault-Tolerance Applied to Unmanned Aerial Vehicles

Comparing the reliability of UAVs to similar manned aircraft is a logical measure of their performance. For the time being, human pilots far exceed the capability of unmanned systems to react to changes in vehicle response. The goal of this research is to achieve a level of reliability in UAVs that exceeds the capability of manned systems and is only limited by the physical constraints of the vehicle. Analyzing the response of a human pilot to in-flight emergencies is a natural starting point for constructing a more reliable unmanned system.

In the event of an emergency, human pilots act on experience and training to execute a prescribed emergency procedure. Successful execution of that procedure requires first, correct diagnosis of the malfunction and second, knowledge of a prescribed procedure to restructure the system. In a simple example, a pilot reacts to an over-temperature in the #2 engine– "Shutdown engine #2 and land as soon as practicable." Successful completion of this procedure, however, entails much more than cutting fuel flow to the bad engine. The pilot must adjust control inputs and path planning so that the vehicle does not exceed its degraded capability. He/she may have to execute emergency procedures or maneuvers not conducted under normal operating conditions. To operate reliably the pilot must adapt to changing conditions and make life-saving decisions.

By their nature, unmanned aircraft present many challenges for fault-tolerance not found in manned systems. The most comprehensive sensor suites on UAVs cannot match the awareness that a pilot provides his aircraft. This lack of awareness often slows detection of a hazardous condition, but also affects the operation of the vehicle after the onset of the fault. State of the art vision based systems cannot execute an emergency landing to an unknown location. In the case of unmanned rotorcraft, conducting an autorotation to even a known and well defined landing area has not been achieved. Another characteristic of UAVs is a lack of redundancy; weight and cost savings often dictate the omission of redundant components. Overcoming these UAV specific challenges and achieving a high level of reliability requires that control systems in unmanned aircraft emulate the following procedures of human pilots:

1. Quickly and accurately diagnose hazardous conditions.

2. Restructure the aircraft in accordance with a prescribed emergency procedure.

3. Adapt control inputs to accommodate for the degraded control response of the aircraft.

4. Constrain flight paths within the capability of the degraded system.

5. Decide on a course of action:

    - Land as soon as possible - land at the nearest suitable landing area (e.g. open field) without delay.
    - Land as soon as practicable - land at a suitable landing area with primary consideration given to the urgency of the emergency.
    - Continue the mission in a degraded mode.

These procedures closely parallel the functions of Clements' fault-tolerant control architecture. The first item corresponds directly with the fault detection and identification component. The restructuring considered in the second item is accomplished in the high level of the Clements' hierarchy during fault isolation. The redistribution controller component in the middle level conducts additional system restructuring. The third item corresponds with the low-level gain controller from Clements' architecture although adaptation may also occur within the low-level or local controllers. In some cases, the third item may necessitate switching to entirely different local controllers. Achieving the fourth item above requires augmentation to the Clements' architecture. His architecture does include a set-point controller, but that component is designed to produce new trim values for the local controllers, not a continuous flight path. Generating a flight path that the impaired aircraft can track is a more difficult, yet achievable, task. The fifth task requires a higher level of autonomy that is not accounted for in Clements' architecture although it could be expanded vertically to accommodate such functionality.

1.4 Plant Descriptions

The fault-tolerant control architecture developed in this research was implemented on two unmanned helicopters: the GTMax and the Renegade UAV. Actual flight testing was conducted on the GTMax. Although the research includes no results on fixed-wing aircraft, it is applicable to unmanned aerial vehicles in general– fixed or rotary-wing. The two helicopters used in testing differ greatly in both size and complexity.

Figure 2. GTMax in flight.

Figure 3. Renegade UAV, a derivative of the Robinson R22.

1.4.1 GTMax

The GTMax (Figure 2) is an automated version of Yamaha's RMax, a remotely controlled helicopter weighing 128 pounds empty with a 10-foot rotor diameter. The rotor system includes a fly bar, and its main rotor blades are rigidly mounted to the mast. Modifications to the base airframe include the addition of two flight computers, an inertial measurement unit, a differential global positioning system, a magnetometer, a sonar, multiple datalinks, and an optical rotor RPM sensor. The optical RPM sensor offers greater precision than the unmodified RMax sensor. The aircraft weighs approximately 160 pounds in its test configuration. The GTMax was a primary research platform for the Defense Advanced Research Projects Agency (DARPA) Software Enabled Control (SEC) [36] and Heterogeneous Urban Reconnaissance, Surveillance and Target Acquisition Team (HURT) projects.

1.4.2 Renegade UAV

The Renegade UAV (Figure 3), operated by the Boeing Company, is a derivative of the Robinson R22 [3]. Researchers constructed it as a surrogate for the A160 Hummingbird. Unlike the GTMax, the Renegade has a teetering rotor system and no fly bar. The Renegade weighs approximately 1350 pounds with a 25-foot rotor diameter. It is the primary research platform for the SEC Renegade program, an extension to the SEC program intended to apply SEC technologies to a full-scale unmanned helicopter. The experiments supporting this research were conducted in a Renegade simulation environment developed at Georgia Tech [24]. The same simulation environment was used in preparation for actual flight tests during the SEC Renegade program. The vehicle model included in the simulation was validated by comparison to actual flight data. Additional information on the SEC Renegade program is available in [24].

1.5 Overview

Chapter 2 provides an overview of the current technologies in fault-tolerant control with an emphasis on the application of reconfigurable flight control algorithms to UAVs. Chapter 3 introduces the fault-tolerant control architecture. Fault detection and identification is also

described in Chapter 3. The next four chapters provide additional detail on individual components of the fault-tolerant control architecture. Chapter 4 discusses adaptive neural networks for reconfigurable flight control and control restructuring on rotorcraft. It includes flight test results on the GTMax. Chapter 5 describes the active system restructuring component in detail and provides additional flight test results on the GTMax. Reconfigurable path planning is the topic of Chapter 6; it includes simulation results for the GTMax. Chapter 7 pertains to mission adaptation and includes simulation results on the Renegade UAV as well as the GTMax. Finally, Chapter 8 provides conclusions and recommended extensions to this research.

CHAPTER 2

BACKGROUND

The pursuit of improved safety and reliability in the aerospace industry has produced several approaches for fault-tolerant control. Early results tended to employ robust, non-adaptive control systems. More recent developments in fault-tolerant control have included adaptive controllers which employ combinations of system restructuring and reconfigurable flight control. Again, restructuring implies an online modification to the inter-connections in the control system. In a car, for instance, the driver typically decelerates using the brake pedal, but in an emergency the hand brake and downshifting are alternate methods to slow the vehicle. Reconfigurable flight control implies online adaptation to the control law. In the simplest case, reconfigurable flight control entails gain scheduling based on the fault condition of the aircraft. Reconfigurable flight control has produced a handful of successful flight tests, mostly on fixed wing aircraft [64, 78]. Research also provides a basis for reconfigurable control of manned and unmanned helicopters [18, 26, 54, 81]. Both control restructuring and reconfigurable flight control follow the intent of the UAV Roadmap when it advocates the development "smart" and "self-repairing" control systems [2].

2.1 Non-adaptive Systems

Non-adaptive control systems provide fault-tolerance without reconfiguration or restructuring. Reliable control is achieved through the selection of a fixed control law that accommodates a set of fault modes while maintaining guaranteed stability properties. Non-adaptive control systems are attractive for two reasons. First, they do not require fault detection and identification (FDI). Second, the fixed control laws are easily implemented [47]. On the other hand, non-adaptive control systems cannot provide an optimal control law for the unimpaired condition and all possible vehicle fault conditions. The result is sub-optimal performance in all modes. The design of so-called reliable control systems is typically

accomplished using H_2 or H_∞ control techniques or using sliding mode control (SMC). Applying SMC to the reconfigurable flight control problem as with other applications has a tendency to produce highly active control signals. Use of a boundary layer on the sliding surface is one method to damp control chattering [35]. Robust control techniques generally result in smoother control signals. A recent H_2 method uses linear matrix inequalities to construct a fault-tolerant flight controller with guaranteed tracking performance [46]. The controller was able to accommodate pre-determined faults at the expense of degradation in the performance of the unimpaired system. This work also demonstrates that the design of non-adaptive control systems becomes complicated even for linear systems. Nonetheless, the idea of accommodating multiple fault modes with a single controller is appealing.

2.2 Adaptive Systems

Assembling multiple individually non-adaptive control systems and switching between them based on system fault condition is a logical advance to the control methodology. The resulting control system can tolerate a large range of faults without degrading the nominal system performance. In addition to multiple reconfigurable controllers, the system requires an FDI capability to switch between the various controllers. These techniques to construct reconfigurable controllers assume that a model of the post-fault system is available during design. The most common methods are the pseudo-inverse method, eigenstructure assignment, and optimal control.

The pseudo-inverse method attempts to recover the performance of the nominal system by computing an approximate matrix inverse. Given a linear system:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{2}$$

where $A \in \Re^{n\times n}, B \in \Re^{n\times m}, C \in \Re^{o\times n}$. The control $u = Kx$ where $K \in \Re^{m\times n}$ results in the following closed loop system:

$$\begin{aligned} \dot{x} &= (A + BK)x \\ y &= Cx. \end{aligned} \tag{3}$$

The post-fault system has the following model:

$$\dot{x}_f = A_f x_f + B_f u$$
$$y_f = C x_f. \tag{4}$$

With A_f and B_f stabilizable, the gain matrix K_f can be computed to stabilize the system,

$$\dot{x}_f = (A_f + B_f K_f) x_f$$
$$y_f = C x_f. \tag{5}$$

The pseudo-inverse method selects K_f to minimize:

$$J(K_f) = ||(A + BK) - (A_f + B_f K_f)||_F \tag{6}$$

where $||.||_F$ indicates the Frobenius norm [29]. This choice of K_f seeks to bring the dynamics of the impaired system close to those of the unimpaired system. The solution to Equation 6 is found using the pseudo-inverse of B_f,

$$K_f = B_f^+ (A + BK - A_f) \tag{7}$$

Difficulties can arise when B_f is not of full row rank. Gao and Antsaklis present a method to generate a K_f that maintains stability of the closed loop system in this case [29]. By integrating FDI into his control design, Bošković advanced the pseudo-inverse methodology. His reconfigurable flight controller applies an implementation of multiple models, switching and tuning (MMST) [12]. The system includes a set of models of the plant dynamics with a fixed pseudo-inverse controller for each model. The controller selects the model that most closely approximates the actual plant dynamics and activates the corresponding controller. Assuming that the actual plant dynamics are sufficiently close to at least one of the models; that each model has at least one stabilizing controller; and that proper switching between the models occurs, the system will stabilize the plant. By applying certain constraints to the switching logic, the designer can ensure proper switching.

Eigenstructure assignment is another method to generate a gain matrix, K_f, that stabilizes the post-fault system (Equation 4). This method is based on the notion that the

response of a closed loop linear system is determined by the location of its eigenvalues and the direction of the associated eigenvectors. K_f is chosen to locate the eigenvalues of the closed loop system at the desired locations (or in desired regions). Any remaining degree of freedom in the gain matrix, K_f is used to minimize the distance between the achieved eigenvectors and a desired set of eigenvectors [42, 71, 85]. This minimization problem is more complex than the pseudo-inverse method, but it achieves a better response from the impaired system particularly when an explicit pseudo-inverse for B_f does not exist. Using eigenstructure assignment in a MMST reconfigurable flight controller is, of course, also possible [86].

Successful implementation of both the pseudo-inverse method and eigenstructure assignment requires the control designer to address control saturation. Matching the performance of the nominal system is often not possible in the presence of a fault. Attempting to recover the nominal performance in this case will result in control saturations. Of course, the post-fault system cannot provide a desired linear response in the presence of control saturations. Bodson suggests four command limiting techniques that recover a degree of stability in the event of control saturation [8]. Implicit model following (IMF) is another method to avoid control saturations. Under IMF control, the aircraft tracks a reference model specified by the control designer. In an adaptive control scheme, the designer can designate different reference models for each fault mode. The reference models can be chosen to reduce or avoid control saturation. In one instance of this control methodology, a reconfigurable flight controller employing model following and the pseudo-inverse method demonstrated fault-tolerance on a tandem rotor helicopter simulation [18].

Optimal control techniques offer a more refined method to alleviate control saturation in the post-fault system. Model predictive control (MPC) generates optimal control inputs to minimize a cost function, for instance the integrated tracking error. The designer can implement various constraints during the optimization including position and rate saturations on the control inputs. Receding horizon MPC produces an optimal control sequence for a finite receding time interval. The first control in that sequence is applied to the system. The primary drawback of MPC is the computation required to solve the optimization problem.

In [10], Bošković details a MMST strategy for a fault-tolerant reconfigurable controller that incorporates MPC. The work parallels [12] with model predictive controllers replacing the pseudo-inverse controllers.

All the approaches to reconfigurable flight control discussed thus far have included a discrete set of controllers and an FDI capability. The MMST architecture discussed above employs one method for FDI, but it requires off-line modeling of system fault modes. Optimal filters offer another means for detecting incipient faults on UAVs such as icing [49]. Sequential Monte Carlo methods or particle filters were also applied to FDI on UAVs [75]. The Fault-Adaptive Control Technology (FACT) employs hybrid bonded graphs to model individual components of the system [4]. This method has been applied to manned aircraft as well as unmanned rotorcraft. Another technique that has been successfully applied to rotorcraft employs static neural networks which are trained off-line using previous flight and simulation data [21, 22]. System identification, which is discussed in Appendix A, can provide additional information for the FDI process [25].

A separate variety of adaptive reconfigurable flight controllers achieves fault-tolerance without dependence on FDI. These controllers typically include a single adaptive controller that can accommodate a range of disturbances, including fault modes, without switching. Because these controllers preclude switching, the interaction between the FDI algorithm and the reconfigurable controller is no longer a consideration for stability. The adaptation is typically driven by an estimation process that excludes prior knowledge of fault modes. Adaptive controllers of this sort are generally divided into two categories. *Direct* adaptive control systems attempt to learn control parameters. *Indirect* adaptive control systems learn parameters of the plant. They commonly employ system identification to estimate a linear model of the plant. Indirect adaptive control then uses the learned model to form a control law. In a sense, FDI-based reconfigurable flight controllers are indirect adaptive possesses that employ a specific form of system identification. Both direct and indirect adaptive controllers draw information from the response of the system. The process requires *persistency of excitation*. Put simply, the signals of the system must contain a sufficient level of information. Sastry provides an analytical explanation for persistency of

excitation [63]. Some authors designate a third category of adaptive reconfigurable flight controllers, self organizing maps (SOMs) [76]. SOMs use neural methods to learn the best control inputs. The learning process usually includes extensive off-line training, but the result is an intelligent, adaptive reconfigurable flight controller.

Indirect adaptive control is appealing because the control process generates useful information about the plant. Flight path planning, FDI, and flight controllers can all utilize the estimated plant model. Furthermore, all the conventional control methods including the pseudo-inverse method, eigenstructure assignment, and optimal control are available to the control designer. Indirect adaptive control depends on the *principle of certainty equivalence* [45]. Uncertain plant parameters are estimated using a system identification process, and those estimates are used in a control law to stabilize of the plant. Due to certainty equivalence, indirect methods cannot accommodate large uncertainties in the plant parameters. Bošković states that indirect adaptive reconfigurable flight controllers can handle only a very limited class of failures [10]. State of the art system identification methods are able to reduce uncertainty in the plant parameters, and the indirect method is commonly used in fault-tolerant control. Indirect adaptive methods that employ receding horizon MPC are common [56, 76]. In one instance, this methodology achieved fault-tolerance on a vertical take off and landing UAV with positive simulation results. Gutièrrez presented a control methodology called adaptive mode transition control (AMTC) [32, 72, 73]. The AMTC architecture enables smooth transition between multiple modes of operation. Mode transition control allows the controller to retain non-local learning. Each mode has an associated indirect adaptive optimal controller. A clear advantage of the indirect method is the availability of optimal control strategies to the control designer.

Direct adaptive control estimates control parameters, not parameters of the plant. The control parameters are immediately useful in the construction of the control vector. Multivariable adaptive control, adaptive neural networks, variable structure control, and adaptive loop-shaping are some of the adaptive control techniques applicable to the reconfigurable

flight control problem. Of these, adaptive loop-shaping is the most primitive, but its foundation in conventional linear control makes it a simple choice for reconfigurable flight control [34]. Bodson proposed a multi-variable model reference adaptive controller for square systems [7]. He compared three implementations of the controller: indirect adaptive, direct adaptive input error, and direct adaptive output error. All three implementations achieved the control objective, but after comparison the direct adaptive input error controller was selected based on stability considerations. Given the inherent robustness of SMC, applying adaptation to SMC methods is another natural choice for reconfigurable flight control. Two methods of adaptation include boundary layer reconfiguration and variable structure control [11, 67]. Both methods provide a means to improve the performance of a non-adaptive SMC controller.

Direct adaptive neural network control is a proven technique to control non-linear systems. The method was selected for use on the Air Force sponsored Reconfigurable Control for a Tailless Fighter Aircraft (RESTORE) program because it has "the ability to stabilize the vehicle following failures / damage" despite model uncertainties [15]. The RESTORE program employed a direct adaptive neural network flight controller in parallel with online system identification [14, 17]. The system identification estimated control derivatives for control allocation, not for indirect adaptive flight control. Adaptive neural network flight controllers employ adaptation to cancel model inversion errors; fault modes manifest themselves as model inversion errors. By adaptively canceling the model error, the system is forced to track a desired reference model. Update laws for the neural networks with guaranteed stability qualities are available [48]. Johnson developed pseudo-control hedging for adaptive neural network controllers [37]. Pseudo-control hedging (PCH) allows the controller to continue adaptation despite control saturation. Kannan extended PCH to a cascaded inner / outer loop structure for the control of an unmanned rotorcraft [39]. The controller employs adaptive neural networks for dynamic inversion in both the inner and outer control loops. The reference model for the inner loop is modified by control saturations in order to protect the adaptive process. The outer loop reference model is modified by inner loop performance. Proctor compensated for latency in Kannan's controller using

direct compensation and a Smith predictor [61].

Self organizing maps add human-like, neural intelligence to the control process. SOMs are able to remember the response of the system in previous flight conditions. They use this memory to determine the best control for a given situation. Like human pilots, SOM flight controllers require extensive training. Dynamic neural programming was used to generate a reconfigurable flight controller for a combat helicopter [27, 28]. This reconfigurable controller employed the constructed mapping to find control solutions for a fault-impaired aircraft. Recursive pseudo-linear regression and dynamic cell structures are other means to update aircraft neuro-controllers [31, 41]. Dynamic cell structures are an extension of SOMs that allow the network to grow and shrink. The NASA Intelligent Flight Control System (IFCS) project employed dynamic cell structures to learn control derivatives from online system identification [50, 70]. Krishnakumar defined four levels (Level 0 - Level 3) of intelligent control for aircraft [44]. Level 0 describes non-adaptive robust feedback controllers. Level 1 describes adaptive reconfigurable flight controllers. Level 2 describes adaptive controllers that optimize their performance over time. Finally, Level 3 controllers include a higher level planning function that accounts for contingencies. None of the controllers referenced thus far achieve Level 3.

All the adaptive control methods reviewed to this point have employed *reconfiguration*, not *restructuring* to achieve their objectives. Restructuring implies adapting the interconnections that define the operation of a system [19]. This adaptation creates a modified system that achieves its control objectives by less conventional means. For instance, varying the main rotor RPM of a helicopter provides a means for restructuring that can accommodate collective actuator faults [43, 81, 83]. This restructuring connects the engine throttle control to a reconfigurable flight controller that generates a variable set point for the main rotor RPM. Osder addressed similar restructuring strategies in a framework of redundancy management [53]. Fixed wing methods include the use of engine thrust to create a pitching moment; the 1989 DC-10 accident in Sioux City, Iowa was referenced. Another fixed wing method is individual control of ganged control surfaces, typically ailerons. For single main

rotor helicopters, a novel technique for control restructuring involves applying a transformation to the swashplate control inputs [54]. Helicopters constructed with canted tail rotors or stabilators possess additional redundancy for pitch control [26]. Other techniques for restructuring rotorcraft include individual blade control and main rotor blade servo flap control [33]. All these methods for restructuring implement a fundamental change in the way the system operates.

Control allocation is a related problem that exists in systems with over-actuation. Control allocation seeks to find an optimal control when the dimension of the control vector u is greater that the dimension of the output vector y (Equation 2). The primary objective of control allocation is to achieve a desired output from the plant. Considering control saturations, this problem may have infinite solutions, a unique solution, or no feasible solution. In the infinite solution case, control allocation can achieve a variety of auxiliary objectives such as minimized control movement or control deflection. Certain control allocation schemes generate control vectors that enhance system identification [16, 20]. Restructuring commonly produces systems that benefit from control allocation. Variations on the pseudo-inverse method provide a relatively simple method for control allocation [5, 58]. Other methods formulate the problem as a linear or quadratic program and solve it using common techniques such as the simplex or interior point methods [60]. Modeling the problem off-line can reduce the online optimization time [6].

2.3 Flight Path Planning

Most UAV flight controllers include a flight path planner of some kind. Path planners, sometimes referred to as trajectory generators or outer loop controllers, generate a desired flight path for the vehicle. The flight path is a sequence of time-stamped position or velocity commands for the low-level controller. Path planners generally create flight paths that are sufficiently conservative so that tracking is not problematic for the low-level controller. Linear low-level flight controllers generally require flight paths designed so that the aircraft does not encounter significant non-linear dynamics. Low-level flight controllers often contain internal reference models that are generated from their command inputs. The reference

models internal to the low-level controller are not synonymous with the flight path. The flight path is created at a higher level in the control architecture. For an unimpaired vehicle, generation of an appropriately conservative trajectory is not difficult. A stationary linear model is usually suitable to create the flight trajectory. After the onset of a fault, however, the capabilities of the vehicle are partially unknown. This makes the generation of attainable flight trajectories problematic. Pseudo-control hedging allows the aircraft to remain stable in control saturation [40, 39]. Adaptive limit detection and avoidance used alone or in conjunction with PCH can ensure that the aircraft does not exceed complex nonlinear limits [82]. Limit avoidance protects the aircraft from exceeding its capabilities, but neither limit avoidance nor PCH removes the necessity for attainable flight trajectories. The process of generating attainable post-fault trajectories is most easily addressed in two steps: 1) model the effects of the failure, 2) conduct online trajectory generation [52]. The first step entails an online system identification process and/or accessing *à priori* knowledge of the capabilities of the damaged vehicle. The second step entails generating an acceptable flight path given the effects of the fault mode. Switching to a pre-determined linear model for flight path generation after the onset of a specific fault is one implementation of this process [88]. More complex algorithms use online optimization and MPC to generate flight paths [23, 55, 56]. Most recently, a developmental intelligent control architecture for UAVs is planned to incorporate a reconfigurable finite-automaton-based path planning approach [74, 78]. A portion of the technique is computationally expensive and conducted in non-real-time. A variation of this approach stores optimization results that are generated offline in a polynomial neural network which can be accessed in real-time [64]. Flight tests on the Total In-Flight Simulator (TIFS) validated the approach for use on a re-useable launch vehicle such as the X-40A. Johnson presented additional results where fault-tolerant guidance was applied to a re-useable launch vehicle [38]. Nonetheless, the area of post-fault trajectory generation is largely unexplored.

2.4 Conclusions

Despite the vast amount of research that has been conducted on fault-tolerance and reconfigurable flight control for UAVs, several shortcomings still exist. Adaptive neural network controllers are very capable at accommodating certain classes of malfunctions, but other fault modes are not accommodated without system restructuring. Furthermore, the majority of reconfigurable flight control research seeks to recover the performance of the nominal system. Depending on the severity of the fault condition, recovery of the unimpaired performance is not a realistic goal. The current research lacks adaptive, reconfigurable path planning algorithms. The subsequent chapter and the remainder of this research develop a fault-tolerant control architecture with the objective of maximizing the utility of UAVs before and after the occurrence of a fault. Drawing from the literature survey in this chapter as well as the discussions in Chapter 1, the following elements are shortfalls of many otherwise suitable architectures.

A suitable fault-tolerant control architecture should:

1. Not degrade the performance of the aircraft prior to the declaration of a fault.
2. Optimize utilization of the aircraft after the declaration a fault.
3. Provide a means for accommodating known and unknown fault modes.
4. Employ prior knowledge of fault modes to minimize response time after the occurrence of a fault.
5. Develop knowledge of the aircraft condition for higher level control processes.

The architecture developed in this research addresses each of these elements by integrating active system restructuring, multiple adaptive neural network controllers, reconfigurable path planning, and mission adaptation in a single fault-tolerant control hierarchy.

CHAPTER 3

FAULT-TOLERANT CONTROL ARCHITECTURE

The fault-tolerant control architecture developed in this chapter is designed to accommodate multiple fault modes with no degradation to the performance of the vehicle before the occurrence of a fault and minimal degradation to the performance after the occurrence of a fault. The goal of the proposed architecture is to expand the usable envelope of the aircraft towards the physical limits of the degraded airframe. The architecture improves reliability by integrating reconfigurable flight control, reconfigurable path planning, and mission adaptation. An active system restructuring component supports this integration. Clements' hierarchical control architecture serves as a foundation for this architecture, but this architecture specifically addresses UAVs. Figure 4 depicts the objective of the fault-tolerant control architecture in expanding the capability of UAVs. This depiction is based on a graphic first presented by the Draper Laboratory during the DARPA SEC project. However, the SEC graphic did not include Degraded Capability arcs. The degraded arcs shift the performance capability of the aircraft towards the origin based on the severity of the fault mode. The reduction in performance indicated by these arcs complicates the fault-tolerant control process because their location is not definitively known. Prior knowledge about relevant fault modes and system identification provide a means to estimate the degraded capability of the aircraft.

Figure 5 depicts the hierarchical fault-tolerant control architecture designed specifically to achieve the objective depicted in Figure 4. Referring to Figure 5, the asterisk indicates components implemented during previous research as part of the baseline control architecture [39]. The current architecture adds several components and functionalities to the baseline architecture and to Clements' architecture (Figure 1). Reconfigurable path planning and mission adaptation components replace Clements' set-point controller. The active

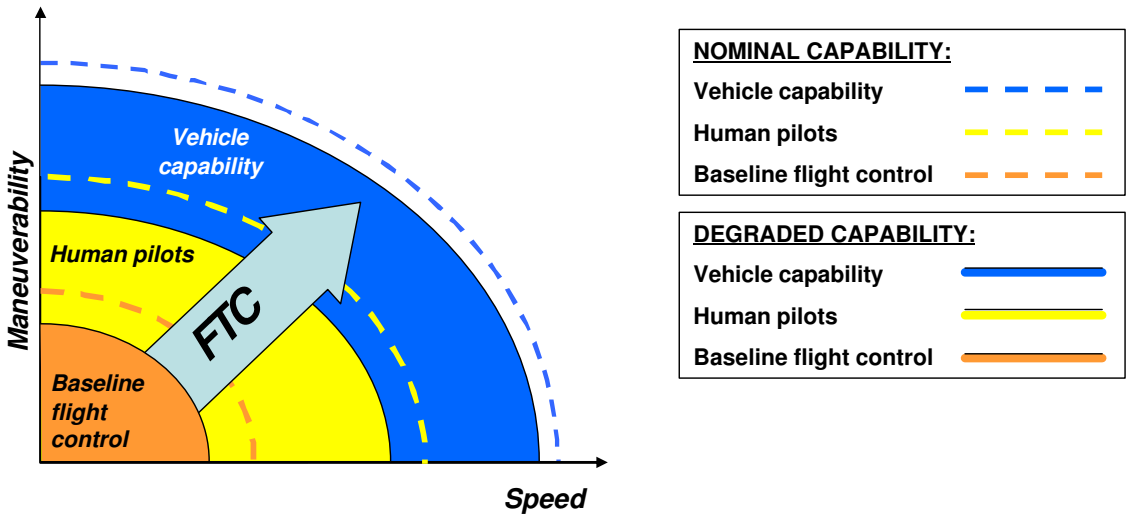

Figure 4. Fault-tolerant control (FTC) objective graphic.

system restructuring component replaces the low-level gain controller and the redistribution controller. The system identification component is an addition, and the reconfigurable flight controllers now augment the baseline controller as local controllers in Clements' terminology. As with the previous architecture, each level of the hierarchy adds autonomy to the vehicle. The architecture is readily expandable vertically as well as horizontally.

The reconfigurable flight controllers and the baseline flight controller reside at the lowest tier of the hierarchy. This layer generates actuator control inputs to achieve a desired flight path. The baseline controller and each of the reconfigurable flight controllers employ adaptive neural networks to provide robustness to uncertainties such as an unidentified fault condition. The reconfigurable flight controllers are not included in the control loop unless activated in reaction to a fault declaration. The low-level controllers are immediately subordinate to the second tier functions.

The second tier receives a sequence of waypoints from the third tier, and it generates a vehicle flight path. It also directs system restructuring in response to fault conditions. When the fault detection and identification component issues a fault declaration, the active system restructuring component sends restructuring instructions to functions at each tier of the hierarchy. In the second tier, the reconfigurable path planning component employs a linear model of the vehicle that is constructed by the system identification component to

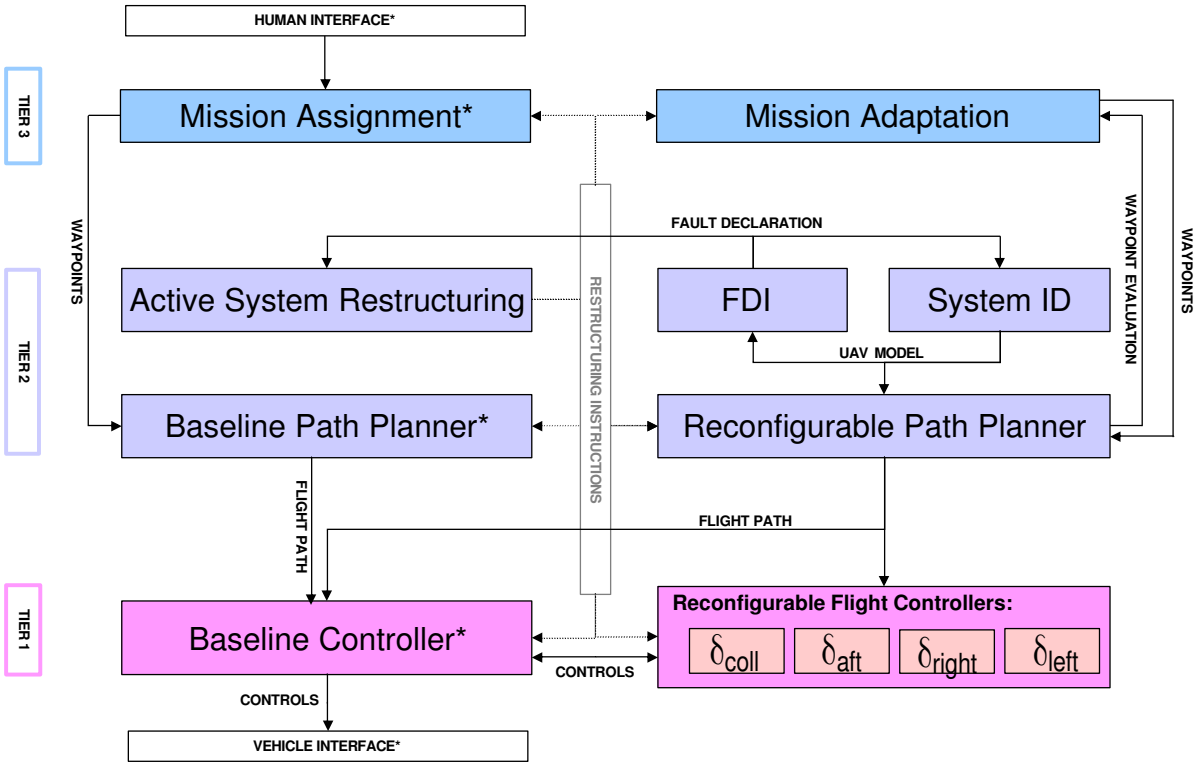

Figure 5. Fault-tolerant control architecture. * indicates components implemented during previous research as part of the baseline control architecture [39].

optimize flight paths for the aircraft.

The human mission commander interfaces with the third tier of the architecture. The human operator typically assigns missions to the vehicle in the form of waypoints. Waypoints are a specified spatial and temporal location for the vehicle. They command the vehicle to be in the proximity of a certain point at a certain time. Missions for the aircraft are formed as a sequence of waypoints, and either the baseline or reconfigurable path planner generates a continuous flight path from a finite set of waypoints. The mission adaptation component adjusts the assigned waypoints based on the condition of the aircraft after the occurrence of a fault. Higher levels of autonomy, not included in the current architecture, could interact with the third tier of the hierarchy.

Analytically, the objective of fault-tolerant control architecture is to optimize the utility of the vehicle:

$$J(M, R) = U(Pe, M, M_{com}) \tag{8}$$

where U is a cost function that quantifies the usefulness of the vehicle to accomplish its mission. U is a function of Pe, actually $Pe(Fm, R)$, which is a measure of the closed loop performance of the aircraft based on the fault mode Fm of the aircraft as well as any restructuring / reconfiguration R applied to the system. Fm is a vector of indicator variables that characterizes the fault modes detected on the aircraft; R is a vector of indicator variables that characterizes all restructuring applied to the system. M_{com} describes the mission assigned to the aircraft, a sequence of waypoints. M allows the fault-tolerant control architecture, specifically the mission adaptation component, to modify the parameters of the assigned waypoints based on vehicle performance, Pe. These parameters include the nominal jerks, accelerations, and velocities used by the path planners. To resolve Equation 8, the active system restructuring component optimizes performance over R. Then, the mission adaptation component optimizes utility over M. In generating this sub-optimal solution, the control architecture assumes that aircraft restructuring and reconfiguration can occur without consideration of the vehicle's mission. Indeed, most restructuring actions taken by pilots in manned aircraft do not depend on the mission profile.

The fault-tolerant control architecture employs active system restructuring to accommodate pre-determined faults that meet the assumptions discussed in Chapter 1. Furthermore, adaptation in all three tiers of the hierarchy allows the architecture to accommodate certain undetected fault modes and even fault modes that are incorrectly identified. The baseline controller and the reconfigurable flight controllers employ a direct adaptive control strategy. The reconfigurable path planning component conducts indirect adaptive guidance. Finally, the mission adaptation component modifies waypoint parameters as needed. Frequency separation between the tiers of the hierarchy prevents interaction between the adaptive processes. The combination of adaptation and active system restructuring for predetermined faults allows the architecture to respond quickly to known faults without losing the capability to cope with unknown fault modes. Flight test results on the GTMax demonstrate that the fault-tolerant architecture can accommodate four separate stuck actuator malfunctions with the effective gain K_{act} reduced to zero (Equation 1).

3.1 Fault Detection and Identification

The success of the fault-tolerant control architecture depends greatly on the success of its FDI routine. Detection and identification of a fault must occur quickly, that is within a few seconds, or the degraded system will readily depart from the flight envelope of the intended reconfigurable flight controller. A small number of false positives are usually acceptable; however, false negatives and mis-identifications can result in the loss of the aircraft. The instability that results from switching between various control strategies is also a concern for FDI. Finally, FDI routines must be robust in design so that neither adverse environmental conditions nor aggressive flight trajectories degrade their performance. Two FDI designs were integrated with the fault-tolerant control architecture, and each demonstrated specific advantages. A state-dependent algorithm detected faults based solely on vehicle flight dynamics. The second was sensor-dependent; it employed sensors to monitor aircraft hardware and detect faults. Of course, a variety of FDI algorithms are available in both categories. The remainder of the section provides specifics on the algorithms that were implemented here.

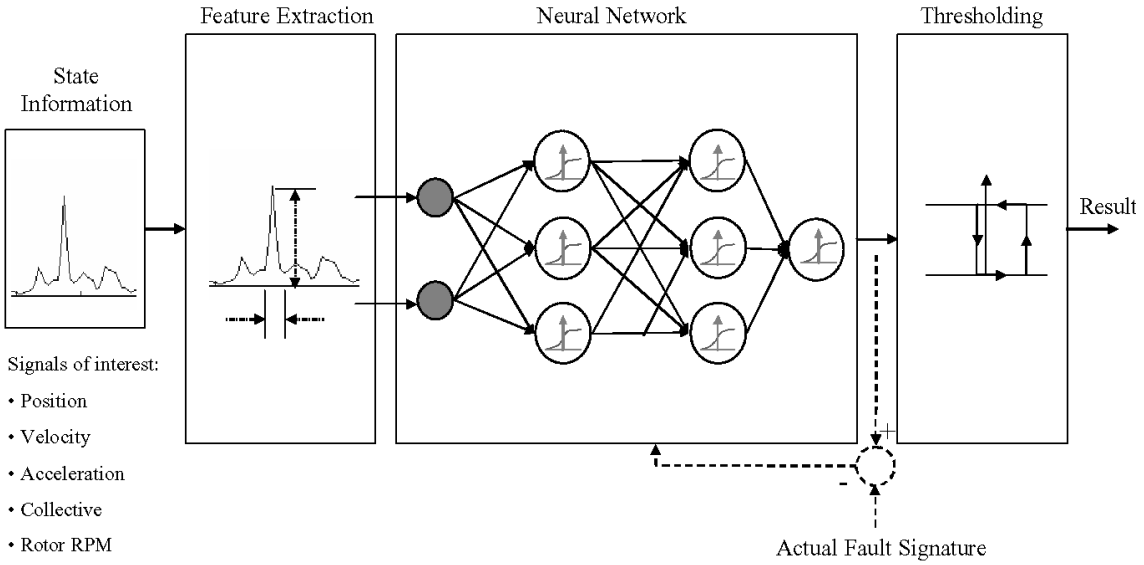

Figure 6. Block diagram of the state-based FDI algorithm as implemented by Clements and Saha [19, 21].

The state-dependent FDI routine employed a neural network trained off-line using actual and simulated flight data. The neural network analyzed the vehicle's state and control vectors to detect the occurrence of a malfunction. In effect, it compared the vehicle response to the expected response given the previous control inputs. Implementation of this technique required no additional sensors. On the other hand, this technique was limited by the richness and the expanse of the training data. For the algorithm to function properly, the training set must include data over a wide range of flight profiles and environmental conditions. This obstacle was surmountable as demonstrated by the flight results.

At the heart of this technique for FDI was a multi-layer feed-forward neural network. The FDI system is depicted as a block diagram in Figure 6. The neural network was trained off-line by back propagation using both simulation and flight data to finalize the weights. The training goal was to minimize the mean square error between the network output and the actual fault flag. The network had an output for each pre-determined fault mode. The input vector was chosen so that it presented a significant correlation with the fault modes under investigation. The input signals were passed through a low pass filter to enhance the signature fault trends. To improve the rejection of false positives and increase confidence in the method, a thresholding condition was imposed on the network outputs. Once sufficiently

trained, the network was used for real-time fault detection. This FDI routine was the result of collaboration with Dr. N. Scott Clements and Bhaskar Saha [19, 21].

If sensors are available on the flight control actuators, sensor-dependent FDI is preferable. Response time is generally improved as well as accuracy in fault identification. Sensor-based FDI has an additional benefit. It directly provides the state of the faulty component. Reconfiguration to accommodate certain actuator malfunctions requires exact knowledge of the faulty actuator's position. Architectures that employ state-dependent FDI can overcome this shortcoming by implementing hardware that locks faulty actuators in a prescribed location once a fault occurs. The Fault-Adaptive Control Technology (FACT) developed at Vanderbilt University [4] employed bonded graphs to detect and precisely identify faults in three flight control actuators on the GTMax. Nagabhushan Mahadevan of the Institute for Software Integrated Systems at Vanderbilt University developed the FACT FDI routine flown on the GTMax.

3.2 System Identification

Unlike the Clements' architecture, the architecture developed in this research includes a separate system identification component. The system identification component produces a model of the open-loop plant for use by other components within the architecture. The low-level flight controllers do not use the output of the system identification component to generate their control laws, but the reconfigurable path planner does use the model to generate flight paths. Likewise, the mission adaptation component depends on the output of the system identification component in order to estimate the capability of the closed loop system. Higher level algorithms rely on closed loop performance estimates to conduct mission planning. Individual components of the architecture can infer a great deal of information on the dynamics of a degraded system from the output of the FDI component. However, at least in the case of UAVs, additional system identification is required to further characterize the response of the impaired system. The architecture developed here allows the designer considerable freedom in selecting the algorithms that reside within the FDI and system identification components. Clearly, the output to the FDI component can aid

the system identification process. Depending on the algorithms implemented, the system identification component can also assist the FDI process. Appendix A outlines a variety of system identification processes for estimating the linear stability and control derivatives that form the A_f and B_f matrices of Equation 4.

3.3 Software Implementation

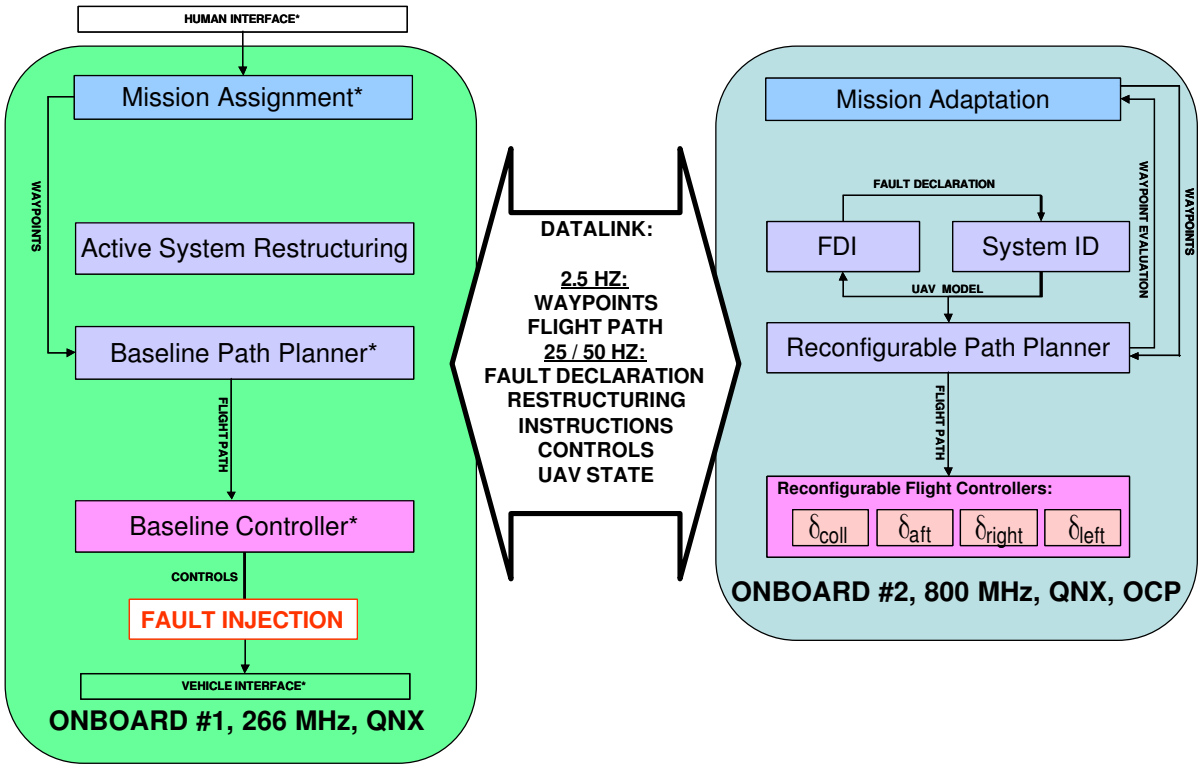

Figure 7. Software implementation on the GTMax.

The fault-tolerant control architecture can reside on a single or multiple onboard processors. The design depicted below (Figure 7) employs the primary and secondary processors installed on the GTMax to divide the computation. In this implementation, the left half of Figure 7 resides on the primary onboard computer, onboard #1. Onboard #1 conducts all the baseline control functions; these functions operate at 50 Hz using the QNX real-time operating system. Additionally, onboard #1 includes a portion of the active system restructuring functionality which interfaces the fault-tolerant control architecture with the baseline control functions in addition to its other functions. One benefit of this implementation is

that the onboard #1 code is largely unchanged by the integration of the fault-tolerant control architecture. The flight path optimization conducted by the reconfigurable path planner is computationally the most expense process. This component as well as the other components depicted on the right half of Figure 7 reside on onboard #2. On onboard #2, the reconfigurable flight controllers, the FDI, and the system identification components operate at 25 Hz. Operating these functions at 25 Hz as opposed the 50 Hz affords the reconfigurable path planner time to operate consistently at 2.5 Hz. Slowing the reconfigurable flight controllers to 25 Hz did not cause a consequential degradation in performance.

3.3.1 Fault Injection

During simulation and actual flight testing actuator faults were injected using software, not hardware. Referencing Figure 7, faults were injected between the baseline controller and the vehicle interface. Faults were simulated by changing K_{act} and/or b using Equation 1. Injecting faults at the bottom of the architecture ensures that every component of the architecture is properly stimulated by the occurrence of a fault.

3.3.2 The Open Control Platform

At times, flight testing on both the GTMax and the Renegade employed the Open Control Platform (OCP) although the fault-tolerant control architecture does not necessitate its use. The OCP is a Common Object Request Broker Architecture (CORBA) based software infrastructure designed to facilitate real-time processing on UAVs [65]. It was developed during the DARPA SEC program by Boeing Phantom Works, the Georgia Institute of Technology, Honeywell Laboratories, and the University of California at Berkeley. The OCP participated in several successful flight tests on various fixed and rotary-wing platforms during the SEC program. Most recently, the OCP enabled successful flight testing of path planning and low-level control algorithms on the Renegade UAV [24].

The OCP provides an open run-time framework that supports multiple control processes running on a single processor or divided across multiple processors. The architecture can extend this interoperability to multiple vehicles as well. A controls application programmer interface (API) raises the level of abstraction so that functions of the CORBA based

architecture are accessible for controls engineers. Using the controls API, control designers can employ the full capability of the OCP to restructure system inter-connections in real-time [80]. This capability makes the OCP particularly suitable for fault-tolerant control.

All fault-tolerant control software that Georgia Tech developed during SEC program including the results presented in Chapters 4 and 5 benefited from the OCP infrastructure. For these flight tests, the onboard #2 processes were compiled in the OCP framework (Figure 7). The OCP enabled seamless real-time communication with the onboard #1 components even though they were not compiled in the OCP framework. The results in Chapters 6 and 7 did not use the OCP. For these results, processing on the onboard #2 machine was scheduled so that the fast processes (25 Hz) were manually prioritized over the slow (2.5 Hz) reconfigurable path planning task without use of the OCP.

CHAPTER 4

RECONFIGURABLE FLIGHT CONTROLLERS

The first tier of the fault-tolerant control architecture (Figure 5), as implemented on the GTMax, includes five reconfigurable flight controllers. This total includes the baseline flight controller which is actually a reconfigurable flight controller capable of accommodating a broad range of faults [15]. The reconfigurable flight controller component includes four additional controllers; each of these is associated with a particular category of fault. The five adaptive neural network controllers combine to extend the fault-tolerance of the architecture across a wide-range of possible malfunctions in any of four flight control actuators. Tail rotor malfunctions, which are not addressed in this research, are discussed briefly in Appendix B.

4.1 Baseline Adaptive Neural Network Flight Controller

The baseline flight control system employs an adaptive neural network and PCH in a feedback linearization scheme to provide precise reference model tracking [37, 39]. The Georgia Tech Unmanned Aerial Vehicles Laboratory, primarily Dr. Eric Johnson and Suresh Kannan, developed and implemented the baseline controller on the GTMax. Extensive flight testing has proven the competence of the nominal controller, which is capable of completing an assigned mission including take off and landing without human interaction. Furthermore, the nominal controller due to its adaptive nature is capable to stabilize the vehicle in the presence of certain classes of fault conditions. The nominal controller is not capable of stabilizing the vehicle when one of the flight control actuators has been completely immobilized or significantly degraded.

The adaptive neural network flight controller conducts dynamic inversion on three control loops, an outer loop, an inner loop, and a throttle loop. The outer control loop controls the vehicle's translational motion (u, v, w) while the inner loop controls angular motion (p, q, r). Under this scheme, the outer loop generates the collective control and an attitude

for the vehicle. The attitude generated by the outer loop is fed to the inner loop which in turn generates the longitudinal and lateral cyclic controls as well as the tail rotor pitch control. The throttle loop adjusts the engine throttle to achieve the commanded main rotor angular rate, Ω_{com}. The neural network allows quick adaptation to the rapidly changing dynamics in all three loops [40].

To further improve controller performance, each loop also includes pseudo-control hedging. PCH adjusts the vehicle reference model to allow adaptation despite control saturation. The inner loop and throttle loop reference models are modified by PCH to account for saturation in each of the vehicle's control actuators. PCH in the outer loop accounts for limitations of the inner loop as well as saturations in the collective actuator. Together the adaptive neural network and PCH make this baseline controller particularly suitable for active system restructuring [40].

4.2 Restructuring to Accommodate Collective Actuator Faults

In many fault scenarios, reconfiguring the baseline control law is sufficient to recover some degree of aircraft controllability. Accommodating other faults requires control restructuring and a completely new control effort. RPM control on a helicopter fits into the latter category. The control vector for a typical single main rotor helicopter includes four inputs: collective, δ_{coll}; lateral cyclic, δ_{lat}; longitudinal cyclic, δ_{lon}; and tail rotor pitch, δ_{tr}. Helicopter vertical thrust is a strong function of both δ_{coll} and Ω, the angular rate of the main rotor. In the nominal state, vertical thrust is controlled by δ_{coll} with Ω held constant. A throttle control loop or engine governor manipulates the throttle, δ_t, to maintain the speed of the main rotor, Ω, at a constant value, Ω_{com}. The converse of this control strategy is feasible with a loss in response time. Variation of Ω requires adding or subtracting rotational energy from the rotor system. To complicate the issue, most helicopter rotor systems include a free-wheeling clutch that enables autorotation but hinders RPM control. Aerodynamic resistance on the rotor blades and hub friction are the only moments available to decelerate the rotor.

The mechanical linkage in many helicopters including the GTMax and the Renegade

UAV are designed without physical actuators that correspond to δ_{coll}, δ_{lat}, and δ_{lon}. A mixing unit typically transforms these virtual stick controls to generate the commands for three swashplate actuators. Details of this transformation are addressed later. Malfunctions to the virtual collective actuator remain of interest because some rotorcraft such as the UH-1 do employ collective actuators and because the handling of collective malfunctions will provide a framework for handling malfunctions in the actual swashplate actuators.

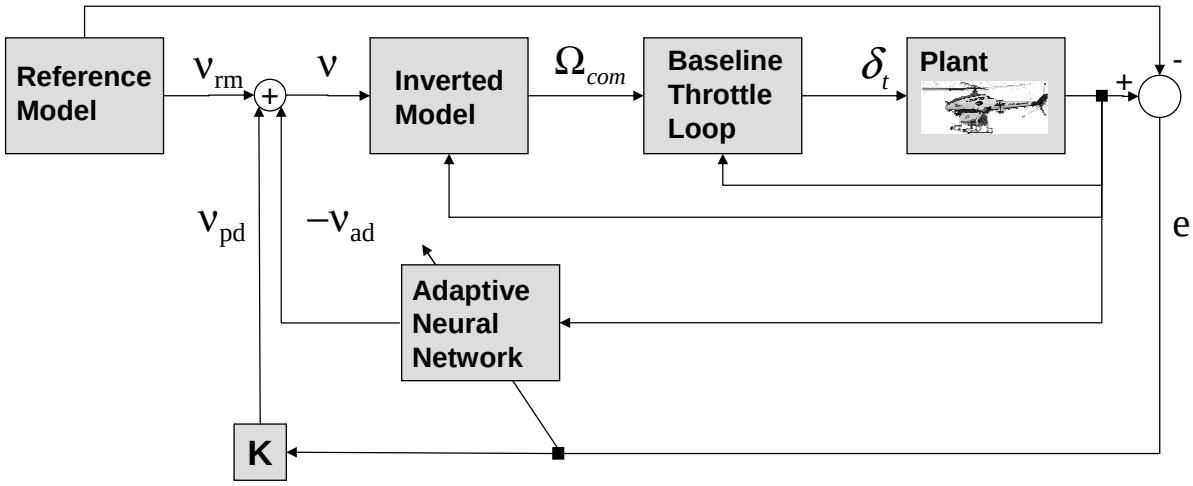

Figure 8. Adaptive neural network reconfigurable flight controller.

Under reconfigurable flight control, the RPM controller (Figure 8) serves as an outer loop for the baseline throttle loop. It generates Ω_{com}, and the baseline throttle loop generates δ_t. The case where δ_{coll} is held stationary represents the most severe class of malfunction. Partially degraded response (reference Equation 1, $0 < K_{act} < 1$) from δ_{coll} provides an improvement over the fully degraded case. Frequency separation allows the RPM controller and the baseline outer loop controller to operate simultaneously with δ_{coll} operating at a higher frequency than Ω. Control restructuring in the event of collective actuator malfunctions does not include any reconfiguration of the baseline controller.

The structure of the RPM controller is identical to the baseline outer loop controllers

with one exception. The RPM controller assumes third order dynamics, not second order:

$$\begin{aligned}\dot{z} &= w \\ \dot{w} &= a_z \\ \dot{a}_z &= j_z(\Omega, \Omega_{com}).\end{aligned} \tag{9}$$

Approximate feedback linearization is achieved using a linear model,

$$j_z = \frac{Z_\Omega}{\tau}(\Omega_{com} - \Omega) \tag{10}$$

where τ is the time constant of the closed loop, baseline throttle loop. Z_Ω is the partial derivative of the aircraft vertical acceleration with respect to the main rotor RPM. Equation 10 is easily invertible to generate a control input, Ω_{com}, from the pseudo-control, ν, where ν is generated in accordance with [39]. In this case, ν drives the plant to emulate the following linear system:

$$\dot{e} = Ae \tag{11}$$

where

$$e = \begin{bmatrix} z - z_{rm} \\ w - w_{rm} \\ a_z - a_{rm_z} \end{bmatrix}. \tag{12}$$

From the characteristic equation,

$$(s + \frac{1}{\tau})(s^2 + 2\omega_n\zeta s + \omega_n^2) = 0, \tag{13}$$

the following A is selected:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -K_p & -K_v & -K_a \end{bmatrix}$$

$$\begin{aligned}K_p &= \omega_n^2/\tau \\ K_v &= \omega_n^2 + 2\omega_n\zeta/\tau \\ K_a &= 1/\tau + 2\omega_n\zeta.\end{aligned} \tag{14}$$

Ω_{com} is constrained by the main rotor limits of the rotorcraft, and pseudo-control hedging accounts for this saturation in the controller design. Under certain assumptions, Theorem 1 in [39] guarantees uniform ultimate boundedness of the tracking error and the neural network weights. Derivation of the control laws required a selection of positive definite matrices, $P \in \Re^{3\times3}$ and $Q \in \Re^{3\times3}$. Appendix C motivates a particular selection for these matrices.

Some of the testing conducted in this research employed an alternate, less sophisticated adaptive neural network controller. That controller, which does not provide the same stability guarantees as the controller described above, is described in Appendix D.

4.3 Restructuring to Accommodate Swashplate Actuator Faults

Active control of Ω_{com} alone is not enough to stabilize a vehicle with a degraded swashplate actuator. However, by applying active control of Ω_{com} in conjunction with a simple transformation to δ_{coll}, δ_{lat}, and δ_{lon} successful compensation for swashplate actuator faults becomes feasible. The resulting system exhibits significant coupling between inner loop cyclic controls and outer loop vertical axis control. Reconfiguring the baseline controller to operate at lower control bandwidths, ω_n, reduces the significance of this coupling.

The swashplate actuators on the GTMax are located at the 3 o'clock, 6 o'clock, and 9 o'clock positions relative to the nose of the aircraft (Figure 9). In the nominal case, the baseline controller generates the virtual controls, δ_{coll}, δ_{lon}, and δ_{lat}, and the following transformation converts them to commands for the swashplate actuators:

$$\delta_{aft} = K_{coll}\delta_{coll} + K_{aft}\delta_{lon} + b_{aft} \tag{15}$$

$$\delta_{right} = K_{coll}\delta_{coll} + K_{lat}\delta_{lat} + b_{right} \tag{16}$$

$$\delta_{left} = K_{coll}\delta_{coll} - K_{lat}\delta_{lat} + b_{left}. \tag{17}$$

Generating this transformation for arbitrary placement of the swashplate actuators is straight forward although additional coupling terms do appear. Research on a high fidelity model of the AH-64 optimized the placement of the actuators for the purpose of fault-tolerance [26].

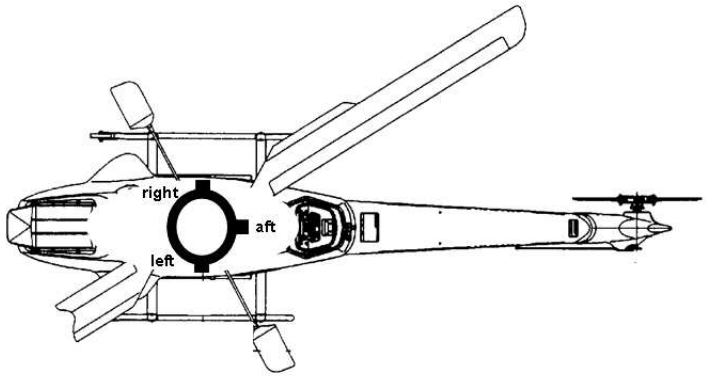

Figure 9. Right, aft, and left swashplate actuators on the GTMax.

In the event of a swashplate actuator malfunction, the transformation is inverted such that the failed position of the swashplate actuator becomes an input and $\hat{\delta}_{coll}$ is an output. $\hat{\delta}_{coll}$ is the collective setting that makes δ_{lat} and δ_{lon} feasible. The redundancy provided by controlling Ω_{com} enables variation of $\hat{\delta}_{coll}$. For the case of a stuck right swashplate actuator, the following equation and Equations 15 and 17 form the inversion:

$$\hat{\delta}_{coll} = 1/K_{coll}[\delta'_{right} - K_{lat}\delta_{lat} - b_{right}]. \tag{18}$$

where δ'_{right} is an estimate of the position of the faulty actuator that is provided by the FDI algorithm.

On the GTMax and the Renegade UAV, RPM control provides the only means for restructuring; however, more complex systems could integrate other control surfaces such as a stabilator, δ_{stab}. In this case, the reconfigurable flight controllers will overwrite the baseline controller's cyclic inputs as well as the collective input with $\hat{\delta}_{lon}$ or $\hat{\delta}_{lat}$ and $\hat{\delta}_{coll}$, respectively. This process is implemented in a manner that permits continued adaptation within the baseline controller.

The following steps describe the process for a malfunction in the aft swashplate actuator (Figure 10. The enumeration below corresponds with the numbering in the figure).

1. Baseline controller generates the virtual controls, δ_{coll}, δ_{lat}, and δ_{lon}.

2. Reconfigurable flight controller generates $\hat{\delta}_{coll}$ and $\hat{\delta}_{lon}$.

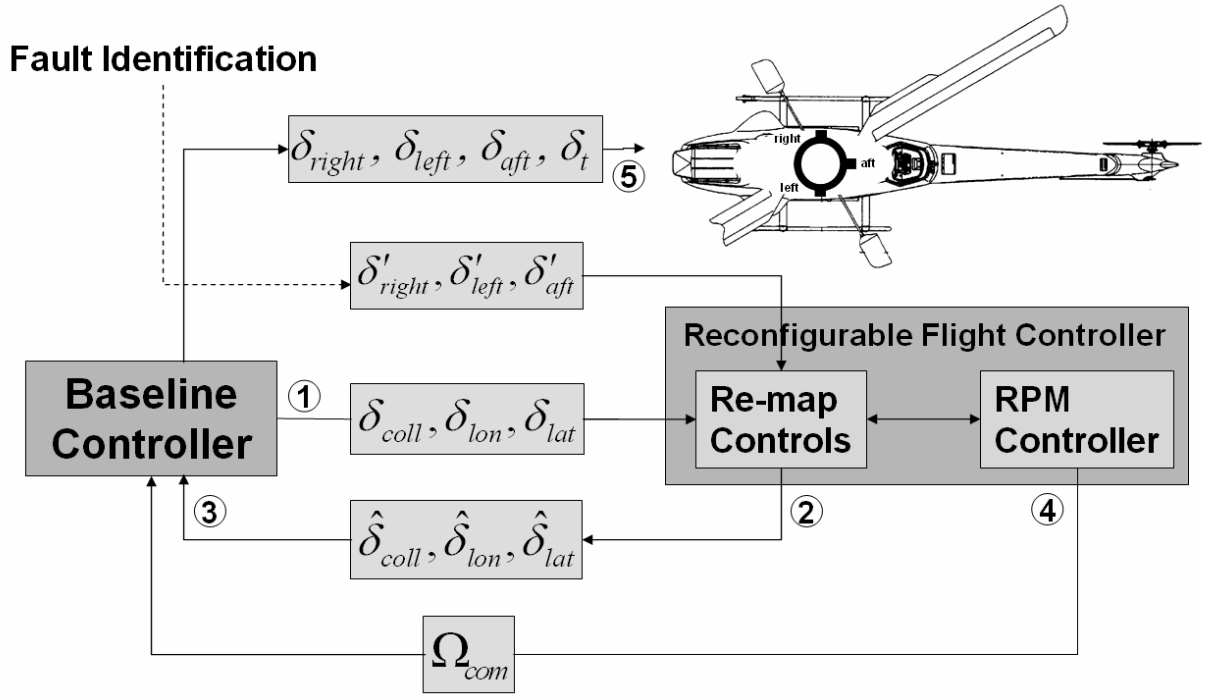

Figure 10. Swashplate actuator reconfigurable flight controllers.

3. $\hat{\delta}_{coll}$ and $\hat{\delta}_{lon}$, are fedback to the baseline controller. The $\hat{\delta}_{coll}$ and $\hat{\delta}_{lon}$ are interpreted as saturations on δ_{coll} and δ_{lon} respectively, and the baseline controller applies pseudo-control hedging.

4. Reconfigurable flight controller employs $\hat{\delta}_{coll}$ to generate Ω_{com} for the throttle loop. In the instance where restructuring includes active control of the stabilator, a second reconfigurable flight controller generates $\hat{\delta}_{stab}$ from the difference $\delta_{lon} - \hat{\delta}_{lon}$.

5. With δ_{aft} immobilized, the derived values of δ_{left}, δ_{right}, δ_t, and perhaps δ_{stab} are applied to the plant.

4.4 GTMax Flight Test Results

Flight tests were conducted on the GTMax to examine the stability and performance of the reconfigurable flight controllers. First, the performance of the stuck collective controller was demonstrated. Next, three flight tests demonstrated the performance of the stuck swashplate actuator reconfigurable flight controllers. During each flight demonstration, the aircraft executed a constant heading box pattern. This pattern was selected to validate the

stability of the controllers while hovering forward, backward, and side-to-side. The flight demonstrations presented in this chapter used a reconfigurable flight controller similar to the one developed above; Appendix D describes the controller used. The results presented in Chapters 6, 7, and 8 employed the controller developed in this chapter. As a means of comparison, the controller employed during flight test operated with $\omega_n = 0.63$ and $\zeta = 0.75$ when combating collective actuator malfunctions. The controller developed above allowed stable operation in simulation with $\omega_n = 1.25$ and $\zeta = 1.0$. Table 2 provides a comprehensive list of the flight tests conducted under this research. Videos of each flight test are available online.

4.4.1 Collective actuator malfunctions

To demonstrate the performance of the collective actuator reconfigurable flight controller, the aircraft was commanded to fly a three-dimensional box pattern with a constant heading of 0 degrees, aligned with the positive x-axis (north). Depicted in Figures 11 and 12, the aircraft starts in the lower front corner at a stationary hover. After 15 seconds, the aircraft initiated the box pattern with a 50-foot climb. Each leg is 10 seconds in duration. The collective was stuck at its hover trim position, b_{trim}, as determined from flight data immediately prior to the demonstration (referencing Equation 1, $K_{act} = 0$, $b = b_{trim}$). For this maneuver, the ω_n and ζ in the RPM controller were set to 0.63 and 0.75, respectively. The largest position error, approximately 3 feet, occurred at the start of the climb, as expected. The use of acceleration feedback resulted in an excessively noisy command signal, Ω_{com}. Future testing should include a filter to protect the throttle actuator from chattering. Plots of the horizontal position errors are provided for comparison to the stuck swashplate actuator flight tests.

4.4.2 Swashplate actuator malfunctions

The subsequent three flight demonstrations validated the performance of the aft (Figures 13 and 14), right (Figures 15 and 16) and left (Figures 17 and 18) swashplate actuator reconfigurable flight controllers, respectively. The aircraft was commanded to execute a level square pattern with constant heading, 270 degrees, aligned with the negative y-axis. For

Table 2. Archive of the online fault-tolerant control flight videos available at `http://uav.ae.gatech.edu/videos/` .

#	Date	Video Clip	Actuator	Comments
1	7-May-04	f030507c1firstRpmWithGunfire.mpg	collective	First flight with PID controller
2	7-May-04	f030507c2rpmConStep10Up.mpg	collective	10 ft climb, PID controller
3	7-May-04	f030507c3rpmConStepDown.mpg	collective	10 ft descent, PID controller
4	7-May-04	f030507c4rpmConClimb70.mpg	collective	70 ft climb, PID controller
5	7-May-04	f030507c5rpmConDown70Guns.mpg	collective	70 ft descent, PID controller
6	7-May-04	f030507d1rpmFDIduringClimb.mpg	collective	70 ft climb, NN FDI, PID controller
7	3-Feb-04	f040203d1_rpmOn.mpg	collective	first flight of ANN controller, stationary
8	3-Feb-04	f040203d2_rpm70ftUp.mpg	collective	70 ft climb, ANN controller
9	3-Feb-04	f040203d3_rpm70ftDown.mpg	collective	70 ft descent, ANN controller
10	3-Feb-04	f040203d4_rpm100ftForward.mpg	collective	100 ft forward, ANN controller
11	3-Feb-04	f040203d5_rpm10fpsBox.mpg	collective	10 ft/s forward, ANN controller
12	1-Apr-04	f040401h1_fdiStuckCol.mpg	collective	hover, NN FDI, ANN controller
13	13-Apr-04	f040413c1_stuckColHover.mpg	collective	hover, ANN controller, windy
14	13-Apr-04	f040413c2_stuckColUp70ft.mpg	collective	70ft climb, ANN controller, windy
15	13-Apr-04	f040413c3_stuckColDown70ft.mpg	collective	70ft descent, ANN controller, windy
16	13-Apr-04	f040413c4_stuckColSquare.mpg	collective	50 ft square, ANN controller, windy
17	13-Apr-04	f040413c5_stuckSbackHover.mpg	aft	First flight of swashplate actuator malfunction, ANN controller
18	13-Apr-04	f040413c6_stuckSbackSquare.mpg	aft	50 ft square, ANN controller, limit cycle
19	25-May-04	f040525e1_stuckCol3dbox.mpg	collective	Climb / descent and square, ANN controller
20	25-May-04	f040525f1_fdiStuckColDown100.mpg	collective	NN FDI, 100 ft descent, ANN controller
21	25-May-04	f040525f2_fdiStuckColUp100.mpg	collective	NN FDI, 100 ft climb, ANN controller
22	9-Jun-05	f040609d1_failedBackSquare.mpg	aft	50 ft square, ANN controller, no limit cycle
23	9-Jun-05	f040609d2_failedRightSquare.mpg	right	50 ft square, ANN controller, no limit cycle
24	9-Jun-05	f040609d3_failedLeftSquare.mpg	left	50 ft square, ANN controller, no limit cycle
25	9-Jun-05	f040609zz1_stuckColBobDown.mpg	collective	NN FDI, 70 ft descent, ANN controller
26	9-Jul-04	f040709c3_fdiNoCollective70ftClimb.mpg	collective	70 ft climb, NN FDI, false positive, ANN controller
27	9-Jul-04	f040709c4_fdiNoSRightBox.mpg	right	FACT FDI, 50 ft square, ANN controller
28	31-Jul-04	f040731b1_fdiBoxWithFailedRightActuator.r	right	FACT FDI, 50 ft square, ANN controller
29	19-Aug-04	f040819b2_stuckRightBox.mpg	right	FACT FDI, 50 ft square, ANN controller
30	19-Aug-04	f040819zz1_stuckCol25fps.mpg	collective	NN FDI, 70 ft descent, 25 ft/s forward flight, ANN controller
31	19-Aug-04	f040819zz1_stuckCol25fps.mpg	collective	NN FDI, 70 ft descent, 25 ft/s forward flight, ANN controller, high collective setting
32	23-Aug-04	f040823d3_vandSRightFailed.mpg	right	FACT FDI, 50 ft square, ANN controller
33	23-Aug-04	f040823f3_fdiStuckCollective.mpg	collective	NN FDI, 70 ft descent, 25 ft/s forward flight, ANN controller
34	24-Aug-04	f040824b1_fdiStuckCollective.mpg	collective	NN FDI, 70 ft descent, 25 ft/s forward flight, ANN controller
35	25-Aug-04	f040825b13_fdiStuckCollective.mpg	collective	NN FDI, 70 ft descent, 25 ft/s forward flight, ANN controller
36	25-Aug-04	f040825c4_vandFdiFailedRight.mpg	right	FACT FDI, 50 ft square, ANN controller

each of the tests, the respective actuator was stuck in its hover trim, b_{trim}, position as determined by flight data immediately prior to the demonstration (referencing Equation 1, $K_{act} = 0$, $b = b_{trim}$). The aircraft started in the lower corner in each plot at a stationary hover. After 15 seconds, the aircraft began the square by moving laterally to the aircraft's right, clockwise. Again, each leg is 10 seconds in duration. The RPM controller was set to operate at $\omega_n = 0.3$ and $\zeta = 1$ for all three demonstrations.

The bandwidth of the baseline controller was also reduced for these demonstrations. Previous flight tests, conducted without reducing the bandwidth of the baseline controller, resulted in a limit cycle characterized by periodic saturations of δ_t and Ω_{com}. For the aft swashplate actuator test, ω_n for pitch control in the inner and outer loops was set to 1.25. For the left and right swashplate actuator tests, the roll control bandwidths were set to 1.25. The controller normally operates at $\omega_n = 2.0$ for the pitch axis and $\omega_n = 2.5$ for the roll axis. This reduction was apparent in the performance of the aircraft in the $x - y$ plane, but it greatly enhanced vehicle performance on the z-axis. The degradation in performance occurs on the y-axis for the aft swashplate actuator fault and on the x-axis for the left and right swashplate actuators.

All four reconfigurable flight controllers performed adequately at a hover. This is notable because normal autonomous landings on the GTMax are slow vertical descents from a stationary hover. Therefore, successful recovery of the vehicle requires stability at a hover. Presumably the swashplate actuator reconfigurable flight controllers would provide a smoother response at greater airspeeds where cyclic control is typically less active; however, they were never activated at ground speeds greater than 10 feet per second. As expected, the collective actuator controller out-performed all swashplate controllers. The right swashplate actuator controller presented a noticeably smoother response than the left swashplate actuator controller. Interaction between the lateral cyclic movements and the tail rotor thrust are the most likely cause for this discrepancy in performance. The aft swashplate actuator demonstrated performance similar to the left swashplate actuator, but drawing conclusions from this comparison is not viable.

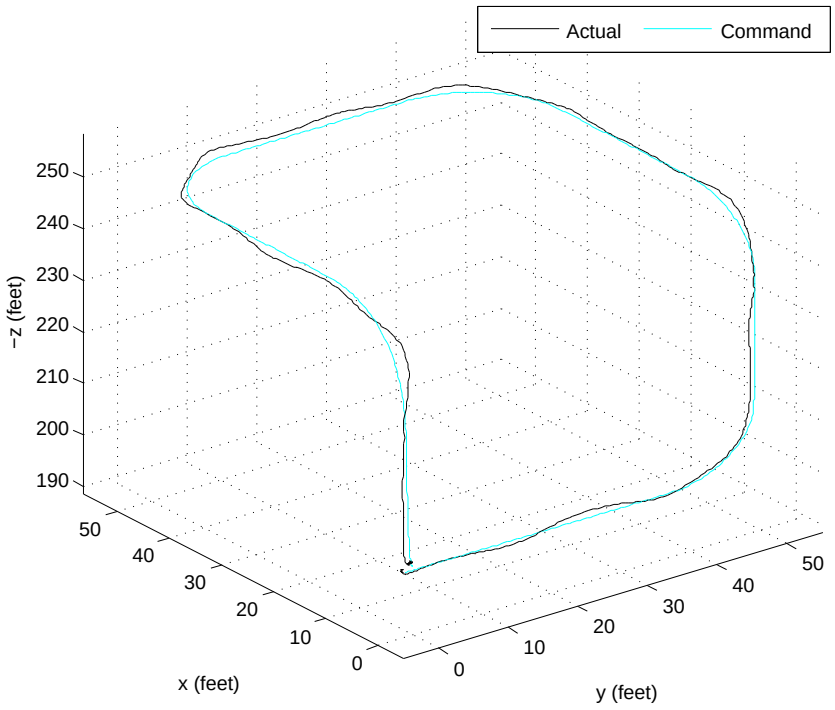

Figure 11. Flight trace with an immobilized collective actuator on the GTMax. Reference test #19 on Table 2.

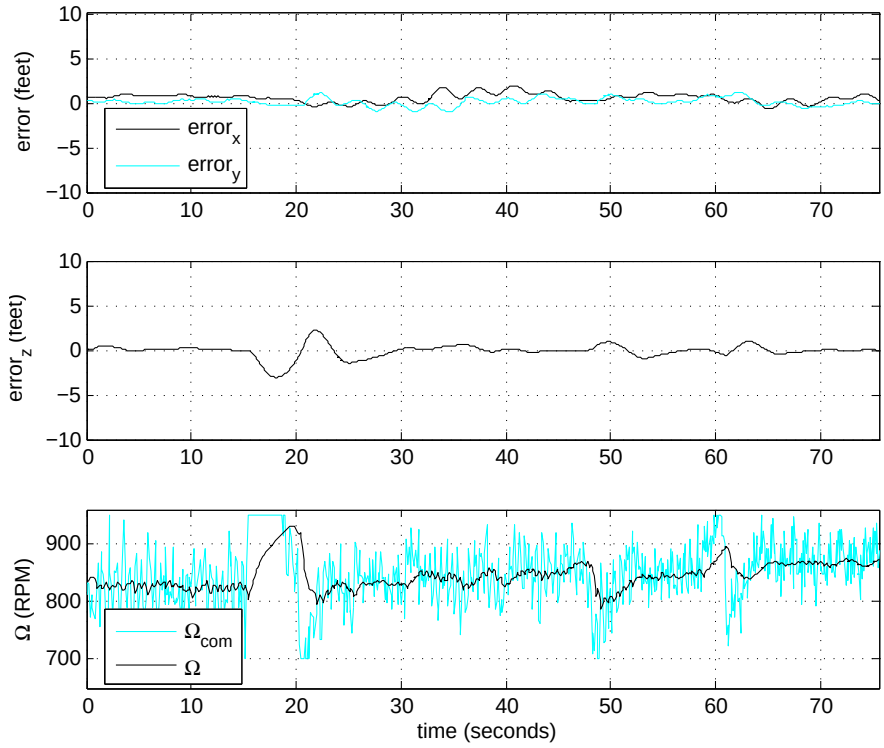

Figure 12. Flight data with an immobilized collective actuator on the GTMax. Reference test #19 on Table 2.

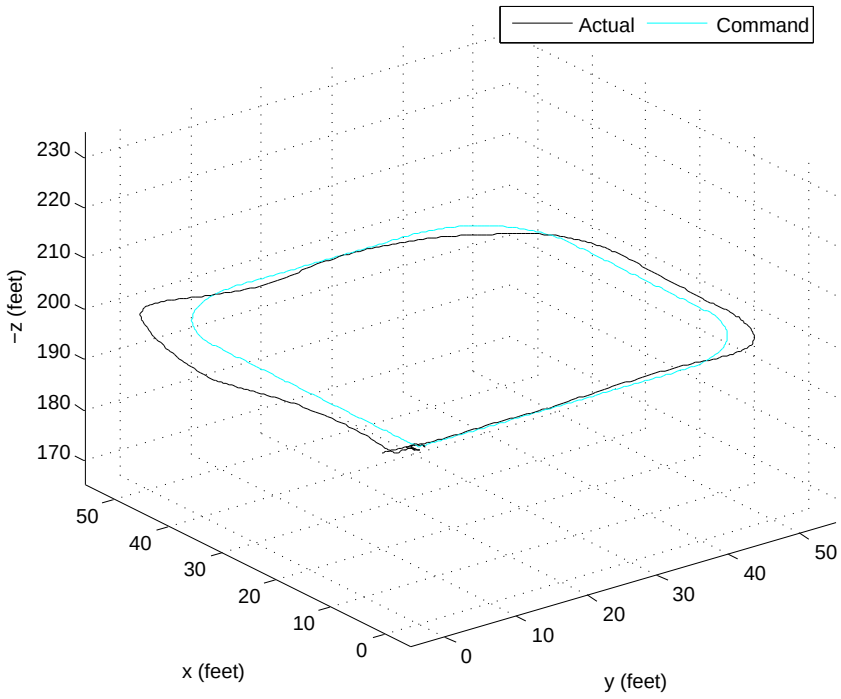

Figure 13. Flight trace with an immobilized aft swashplate actuator on the GTMax. Reference test #22 on Table 2.

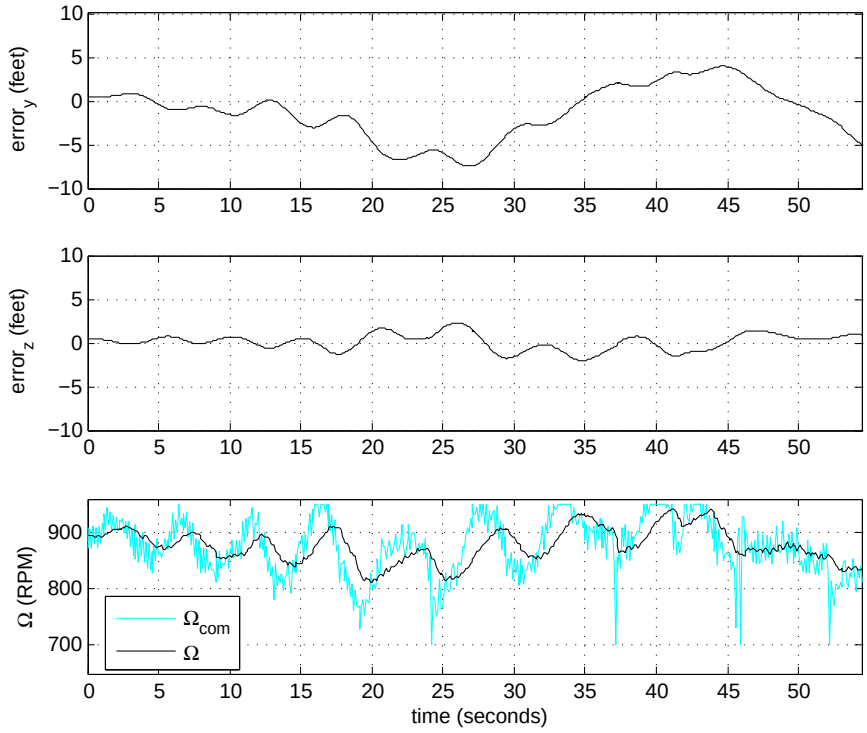

Figure 14. Flight data with an immobilized aft swashplate actuator on the GTMax. Reference test #22 on Table 2.

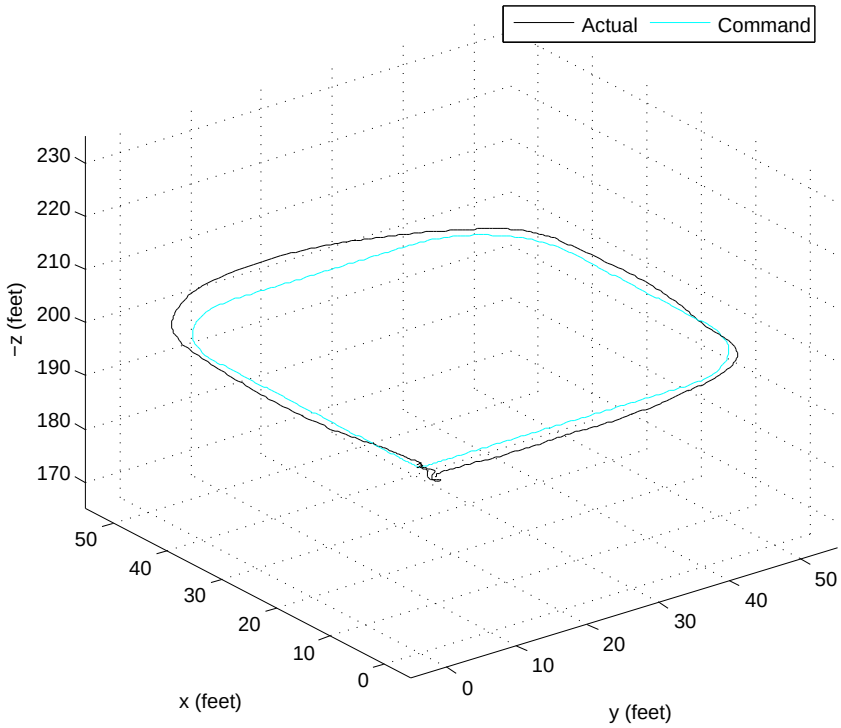

Figure 15. Flight trace with an immobilized right swashplate actuator on the GTMax. Reference test #23 on Table 2.

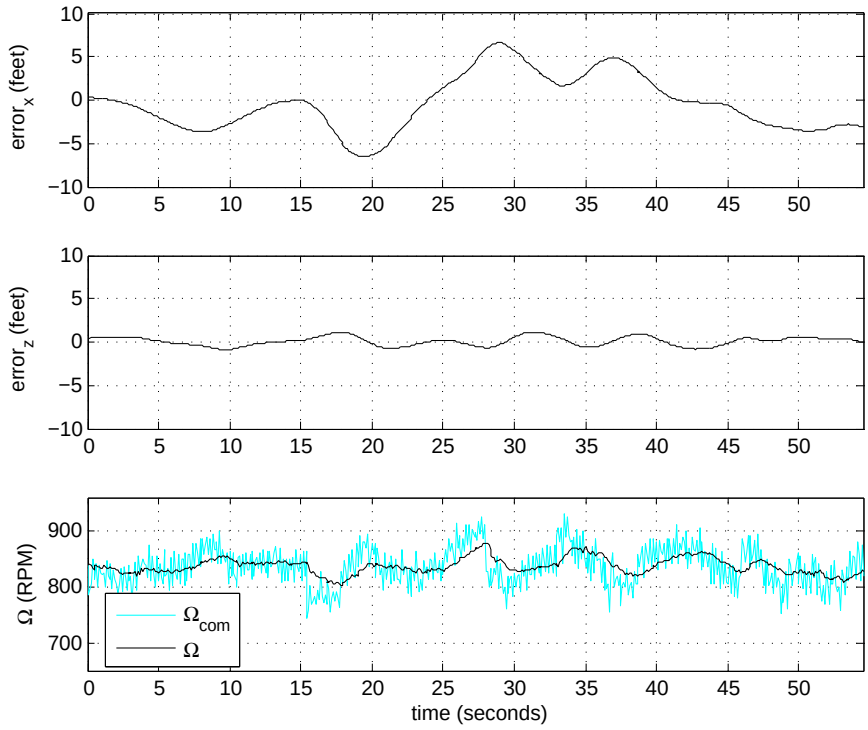

Figure 16. Flight data with an immobilized right swashplate actuator on the GTMax. Reference test #23 on Table 2.

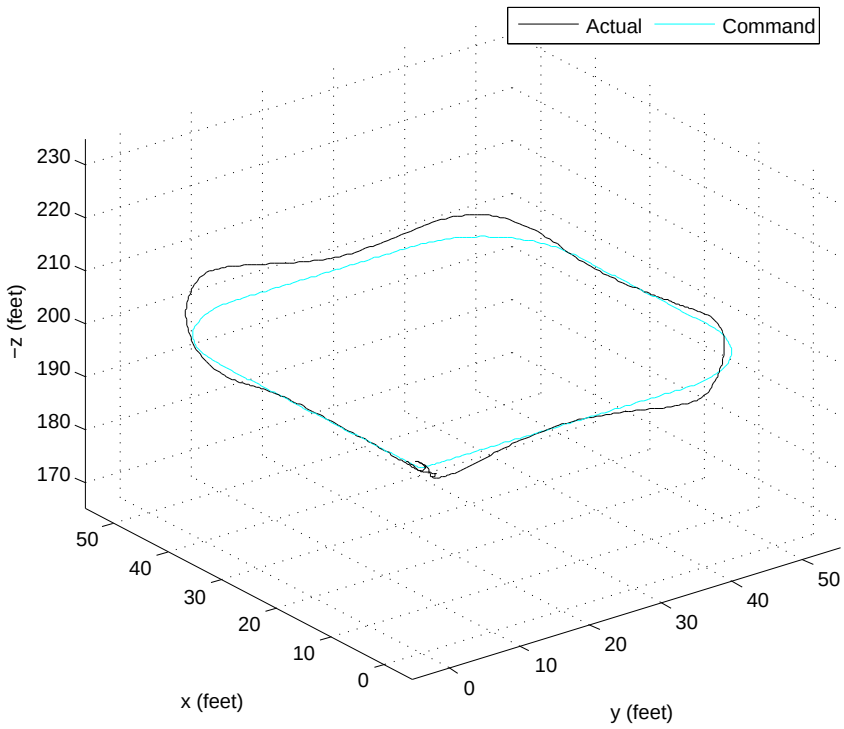

Figure 17. Flight trace with an immobilized left swashplate actuator on the GTMax. Reference test #24 on Table 2.

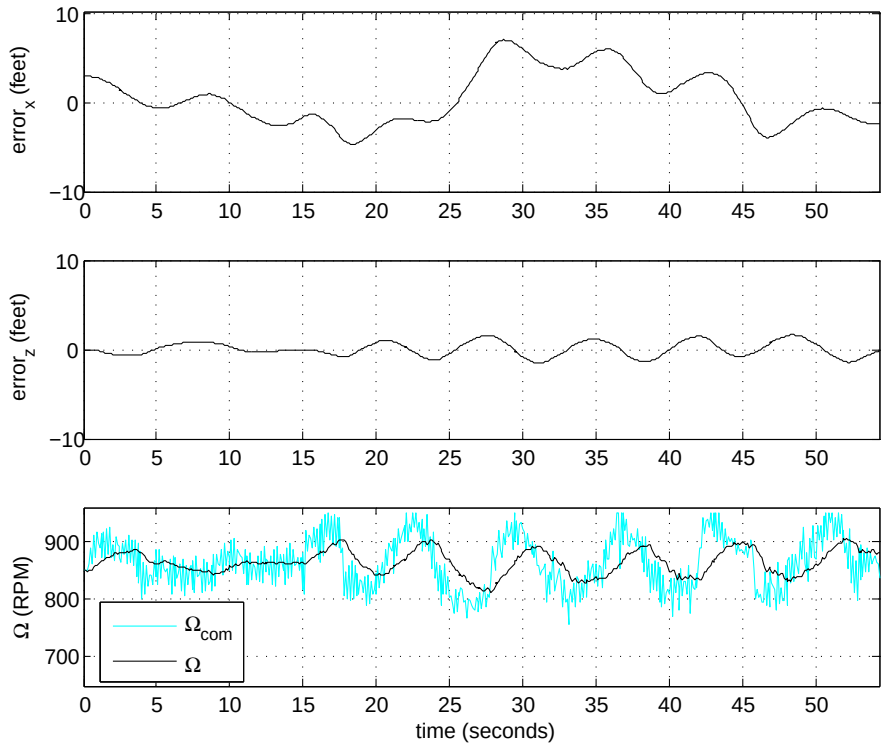

Figure 18. Flight data with an immobilized left swashplate actuator on the GTMax. Reference test #24 on Table 2.

CHAPTER 5

ACTIVE SYSTEM RESTRUCTURING

The active system restructuring component synchronizes the fault-tolerant control architecture's reaction to a fault. It modifies the inter-connections in the control architecture and forces reconfiguration in the control laws. Doing so, it conducts the high-level fault isolation task of Clements' architecture, and it replaces the redistribution controller and the low-level gain controller components from the mid-level (Figure 1). A fault declaration requires a specific response from every component in the architecture. The active system restructuring component issues restructuring instructions to direct these responses (Figure 5). Employing prior knowledge about the structure of the system, it reacts quickly to counter a set of pre-determined faults. Because the low-level controllers are adaptive, a single restructuring / reconfiguration compensates for a range of faults. For instance, a single reconfigurable flight controller accommodates all variations of collective malfunctions (Equation 1). Restructuring serves to maximize the existing control effort onboard the vehicle. This process often necessitates abandoning some of the control objectives of the nominal system.

5.1 Optimization Based Active System Restructuring

In accordance with Equation 8, the objective of the active system restructuring component is to optimize the performance of the vehicle following the occurrence of a fault:

$$J(R) = Pe(Fm, R) \tag{19}$$

where optimization occurs over R, a vector that indicates all restructuring and reconfiguration actions applied to the aircraft. The active system restructuring component can reconfigure the control bandwidths within the baseline and reconfigurable low-level flight controllers in addition to restructuring the control architecture. A simple yet effective

expression for Pe has the following form:

$$Pe = \| e \| + Fm^T W R. \tag{20}$$

The expression includes an error function, $\| e \|$, and a simple cost function. The control designer should designate an expression for $\| e \|$ that achieves his control objectives. One feasible selection is the root mean square position error of the aircraft through the remainder of its mission. Clearly, such a choice for $\| e \|$ requires significant computation for even a modest number of restructuring options. The linear cost term uses the weight matrix W to penalize certain restructuring actions dependent on the fault condition of the aircraft, Fm. The cost term allows the control designer to inject prior knowledge about the system into the performance function.

The following simple example is constructed in the framework of Equation 20. Assume the helicopter's vertical dynamics are governed by the following linear system:

$$\begin{bmatrix} \dot{w} \\ \dot{\Omega} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix} \begin{bmatrix} w \\ \Omega \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} \delta_{coll} \\ \Omega_{com} \end{bmatrix}. \tag{21}$$

where it is desired to maintain Ω at its initial value, $\Omega(0) = 0$. R_Ω is a restructuring indicator; if $\Omega_{com} \neq 0$, $R_\Omega = 1$; else $R_\Omega = 0$. A simple performance function Pe for this system takes the form:

$$Pe = |\ddot{w} - \ddot{w}_{rm}| + |\dot{w} - \dot{w}_{rm}| + |w - w_{rm}| + W_\Omega R_\Omega. \tag{22}$$

Note this choice for the error term,

$$\| e \| = |\ddot{w} - \ddot{w}_{rm}| + |\dot{w} - \dot{w}_{rm}| + |w - w_{rm}| \tag{23}$$

only considers the current time step. The cost coefficient W_Ω penalizes restructuring to prevent unnecessary active control of Ω. Despite the simplicity of this performance function, it enables re-allocation of control effort to accommodate degradation in the response of the collective control input, i.e. a reduction in b_{11}. $\ddot{w}$ is a linear function of Ω_{com} so with $R_\Omega = 1$ reduction of the error term, $|\ddot{w} - \ddot{w}_{rm}|$, is possible.

Analysis of this performance function, Equation 22, is only possible because the effect of varying Ω_{com} is known in advance. This is typical of all Pe functions. In order to optimize

aircraft performance, the control designer must know the effect that each restructuring action will have on the vehicle. For complex systems, the extensive modeling that is required to capture accurately the effect of every restructuring action becomes prohibitive. Furthermore, substantial computation is required to optimize complex performance functions. In either case, look-up tables are an efficient alternative. An emergency procedure database developed off-line using component-based modeling or heuristic judgment can replace an analytical function for Pe. Incidentally, referring the Equation 19, an emergency procedure database can be viewed as a performance function with no error term. The flight test results in this chapter demonstrate use of such a database on the GTMax.

Figure 19 depicts the interaction of the active system restructuring component on the GTMax with the baseline controller and with the reconfigurable flight controllers developed in the previous chapter. Although not depicted in the figure, the active system restructuring component also sends restructuring instructions to the reconfigurable path planner so that it can re-shape flight paths; to the system identification component so that it can use knowledge of the fault to improve its approximation; and to the mission adaptation component so that it can modify the sequence of waypoints if necessary. Table 3 conveys the emergency procedure database employed by the active system restructuring component on the GTMax. The table includes procedures to accommodate tail rotor malfunctions although a reconfigurable flight controller for this malfunction was not developed (Appendix B).

5.2 GTMax Flight Test Results

The results presented in this section demonstrate the integration of two FDI algorithms, multiple reconfigurable flight controllers, and the active system restructuring component as well as all the baseline components. The first two flight tests below employed the state-dependent neural network FDI routine to detect collective actuator malfunctions. The third flight test used Vanderbilt's FACT FDI algorithm to identify a fault in one of three swashplate actuators. Each of the flight demonstrations presented below used the reconfigurable flight controller described in Appendix D. Table 2, located in the previous chapter, provides a comprehensive list of the flight tests conducted under this research.

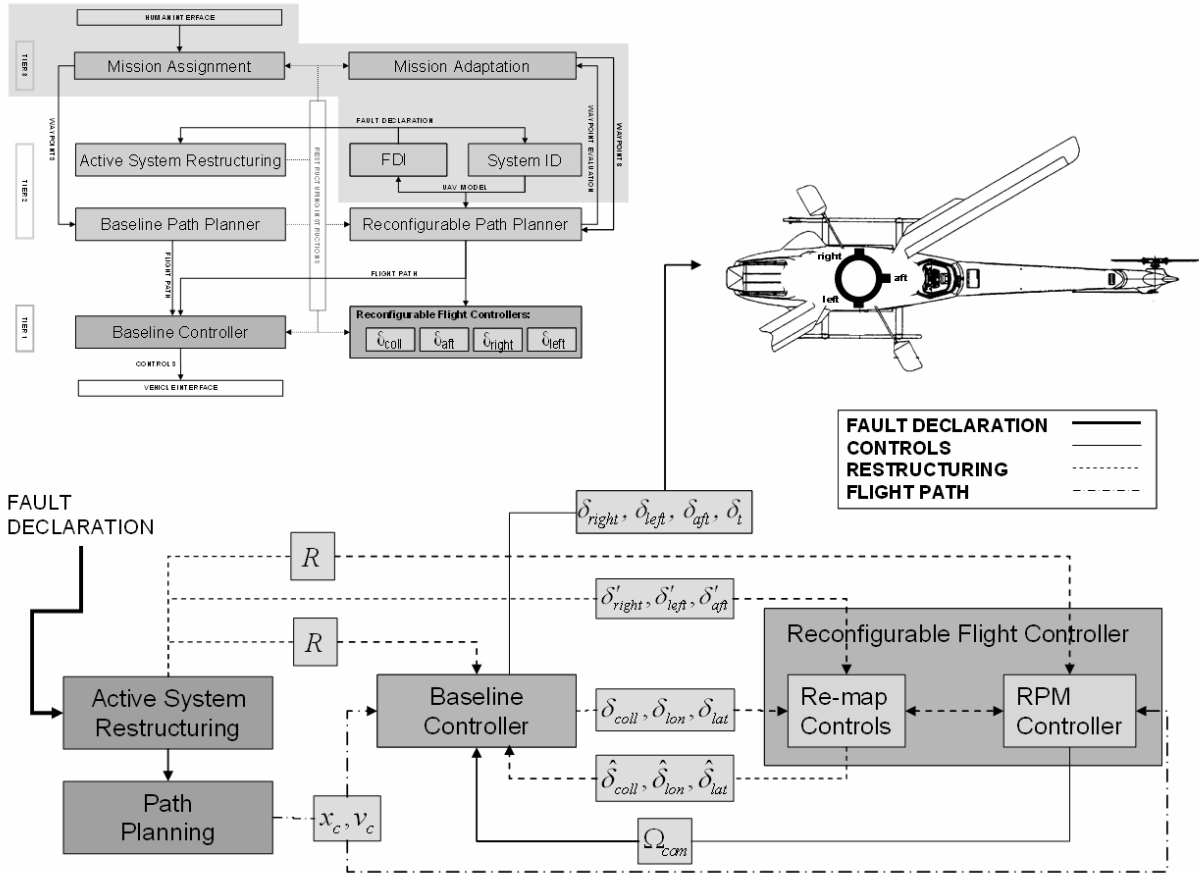

Figure 19. Active system restructuring on the GTMax.

Table 3. Emergency procedure database employed by the active system restructuring component on the GTMax.

Fault	Restructuring	Reconfiguration
Collective Actuator Fault	1. Activate reconfigurable flight controller to actively control main rotor RPM. 2. Activate the reconfigurable path planner. 3. Activate the mission adaptation component.	1. Set reconfigurable flight controller parameters: ω_n=0.63, ζ=0.75*. 2. Switch to the appropriate fault model in the system identification component, and enable adaptation.
Aft Swashplate Actuator Fault	1. Activate reconfigurable flight controller to actively control main rotor RPM. 2. Restructure the baseline controller to enable external control of the collective. 3. Activate the reconfigurable path planner. 4. Activate the mission adaptation component.	1. Set reconfigurable flight controller parameters: ω_n=0.3, ζ=1.0. 2. Set baseline flight controller inner and outer loop pitch control parameters: ω_n=1.25, ζ=1.0. 3. Switch to the appropriate fault model in the system identification component, and enable adaptation. 4. Limit nominal acceleration to 2.5 ft/s/s.
Right/Left Swashplate Actuator Fault	1. Activate reconfigurable flight controller to actively control main rotor RPM. 2. Restructure the baseline controller to enable external control of the collective. 3. Activate the reconfigurable path planner. 4. Activate the mission adaptation component.	1. Set reconfigurable flight controller parameters: ω_n=0.3, ζ=1.0. 2. Set baseline flight controller inner and outer loop roll control parameters: ω_n=1.25, ζ=1.0. 3. Switch to the appropriate fault model in the system identification component, and enable adaptation. 4. Limit nominal acceleration to 2.5 ft/s/s.
Tail Rotor Actuator Fault	1. Activate reconfigurable flight controller to actively control main rotor RPM. 2. Activate the reconfigurable path planner. 3. Activate the mission adaptation component.	1. Set reconfigurable flight controller parameters, appropriately. 2. Reduce baseline flight controller heave mode control parameters. 3. Switch to the appropriate fault model in the system identification component, and enable adaptation. 4. Heading mode aligns aircraft with commanded velocity vector. Limit vertical accelerations.

* The simulation results presented in all subsequent chapters used the reconfigurable flight controller developed in the previous chapter and ω_n=1.25, ζ=1.0.

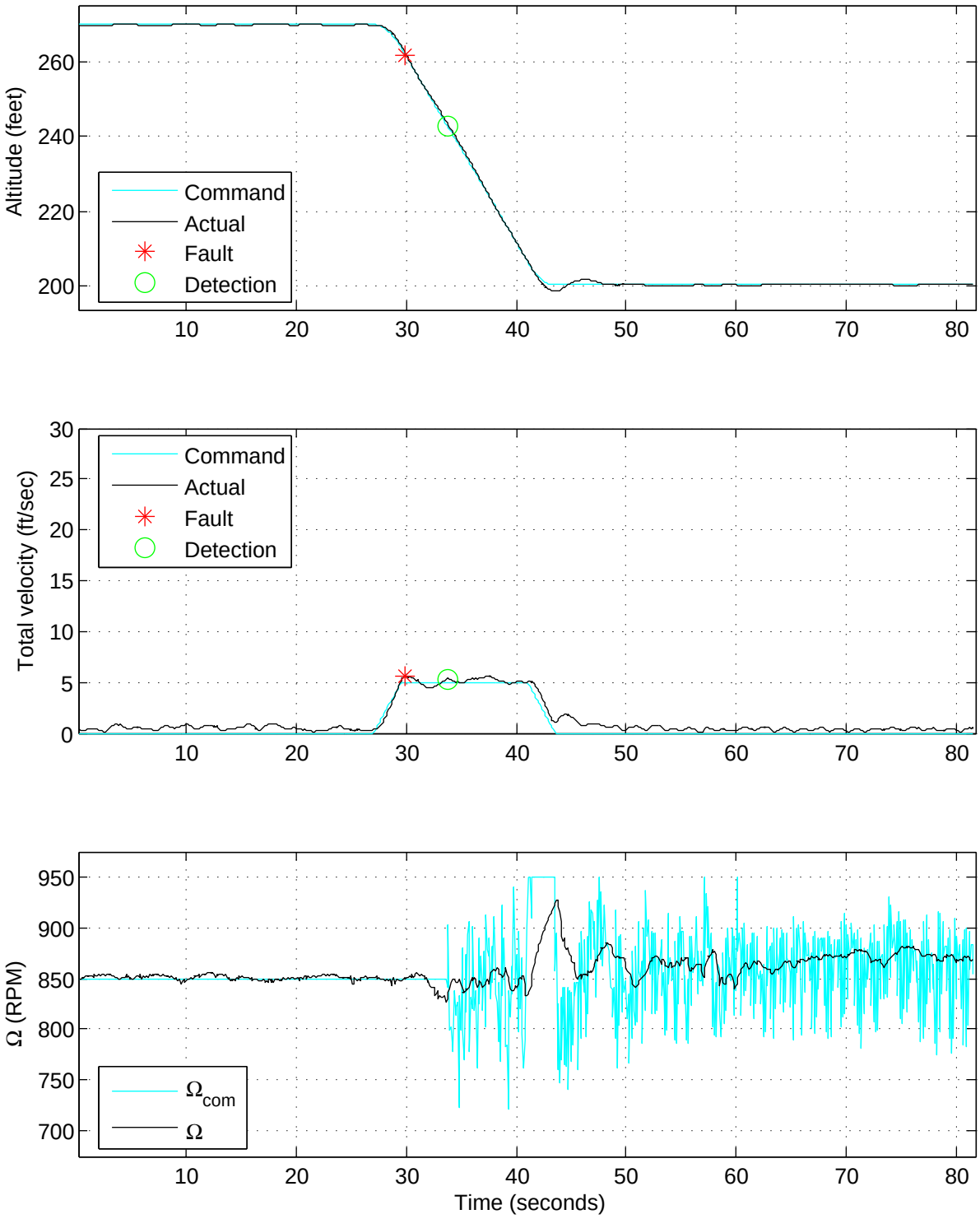

Figure 20. Flight data for an immobilized collective actuator with integrated neural network FDI on the GTMax. Reference test #25 on Table 2.

5.2.1 Actuator malfunctions with state-dependent fault detection and identification

The first flight demonstration was initiated with the UAV in its baseline configuration with no fault applied. The aircraft was commanded to execute a 70-foot descent from a stationary hover. During the descent, the stuck collective fault was applied binding the collective in a typical descent position that was determined from flight data on the day of the flight test. The neural network FDI routine (Figure 6) detected the fault and issued a fault declaration to the active system restructuring component. The active system restructuring component elected to activate the appropriate reconfigurable flight controller. This restructuring compensated for the loss of control authority in the vertical axis, but it abandoned the control objective to maintain a constant main rotor RPM. While the vertical response of the aircraft is degraded, adequate performance from the throttle loop allowed active control of Ω_{com} to stabilize the vehicle. Switching to a reconfigurable flight controller implies a change to the inter-connections of the system. Referencing Figure 20, the descent was initiated at 28 seconds; the fault is applied at 30 seconds; and the fault was detected prior to 34 seconds. Without reconfiguration, the vehicle would not have been able to arrest its descent.

At a later date, subsequent flight tests demonstrated that both the FDI routine and the reconfigurable flight controller can accommodate a collective actuator malfunction while tracking a considerably more aggressive flight path. Again, the aircraft started with a 70-foot descent from a stationary hover. The aircraft descended at 10 feet per second, twice the previous velocity, and accelerated to a forward speed of 25 feet per second. Figure 21 traces the flight path of the vehicle.

Figure 22 presents three time-histories from this flight test. At the bottom of the descent the aircraft slows from 25 feet per second to 20 feet per second. This is the most demanding portion of the demonstration for the reconfigurable flight controller; the main rotor RPM decreases to near 725 (t=42 seconds). During the remainder of the flight test the main rotor RPM was maintained between 780-920. Reconfigurable path planning, introduced in the next chapter, serves to protect the reconfigurable flight controller from overly demanding

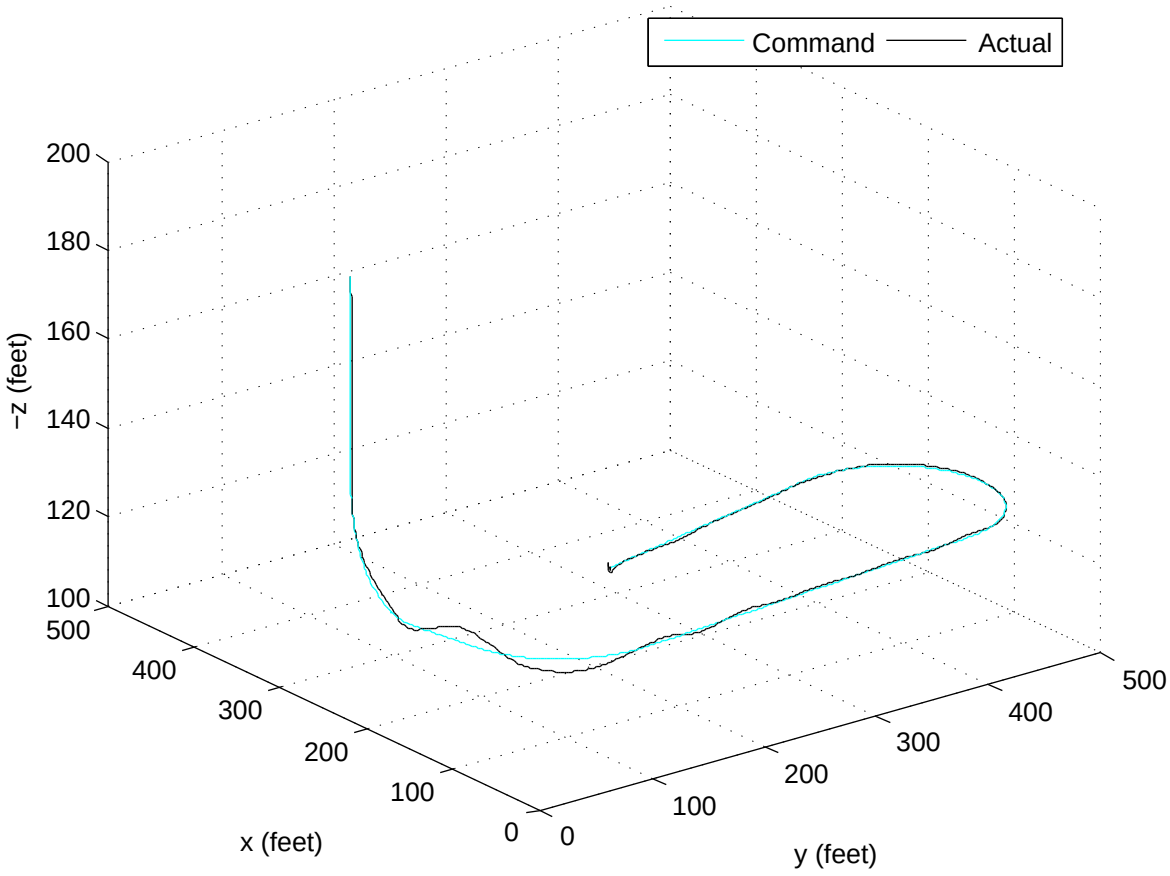

Figure 21. Flight trace for an immobilized collective actuator with integrated neural network FDI in forward flight on the GTMax. Reference test #30 on Table 2.

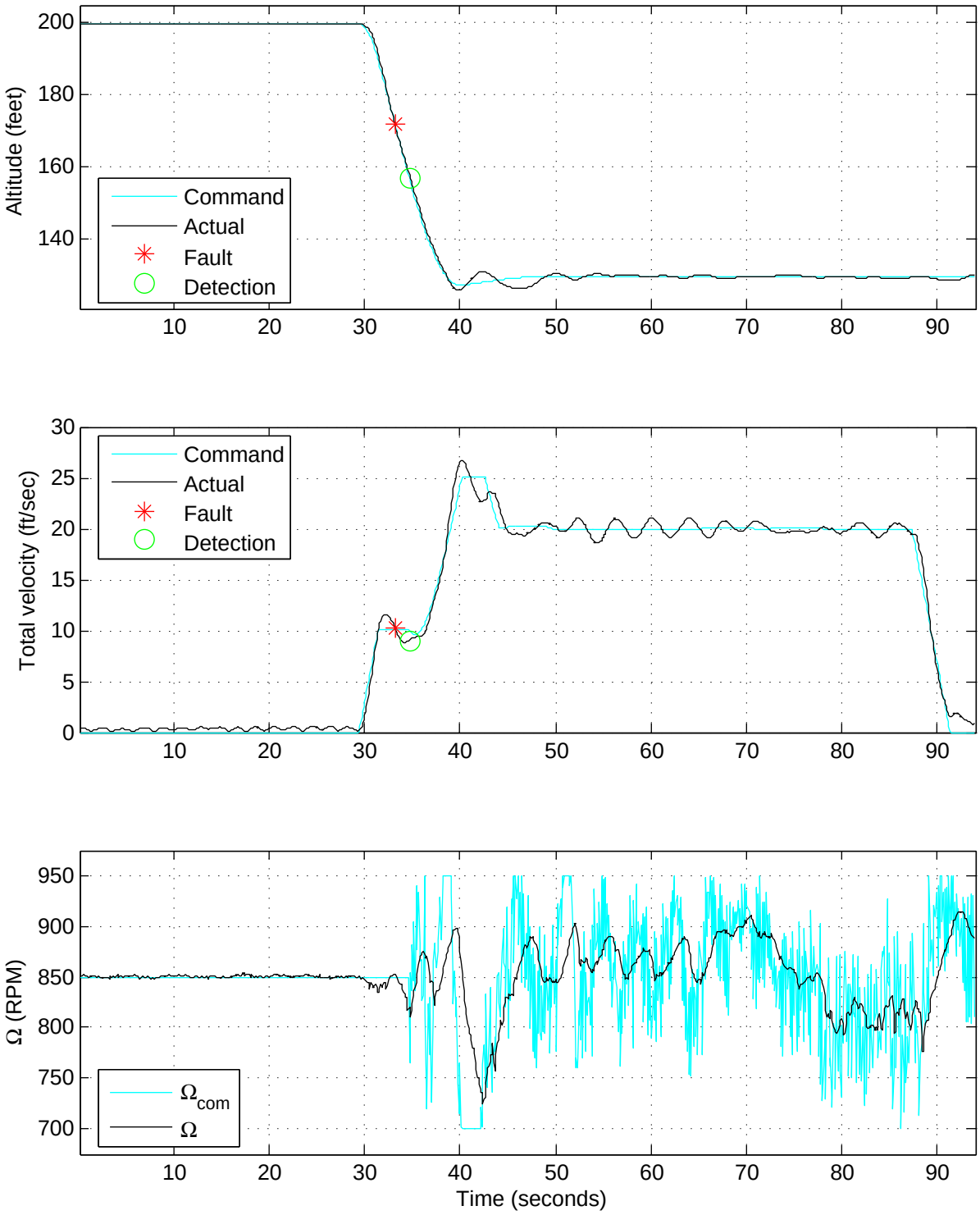

Figure 22. Flight data for an immobilized collective actuator with integrated neural network FDI in forward flight on the GTMax. Reference test #30 on Table 2.

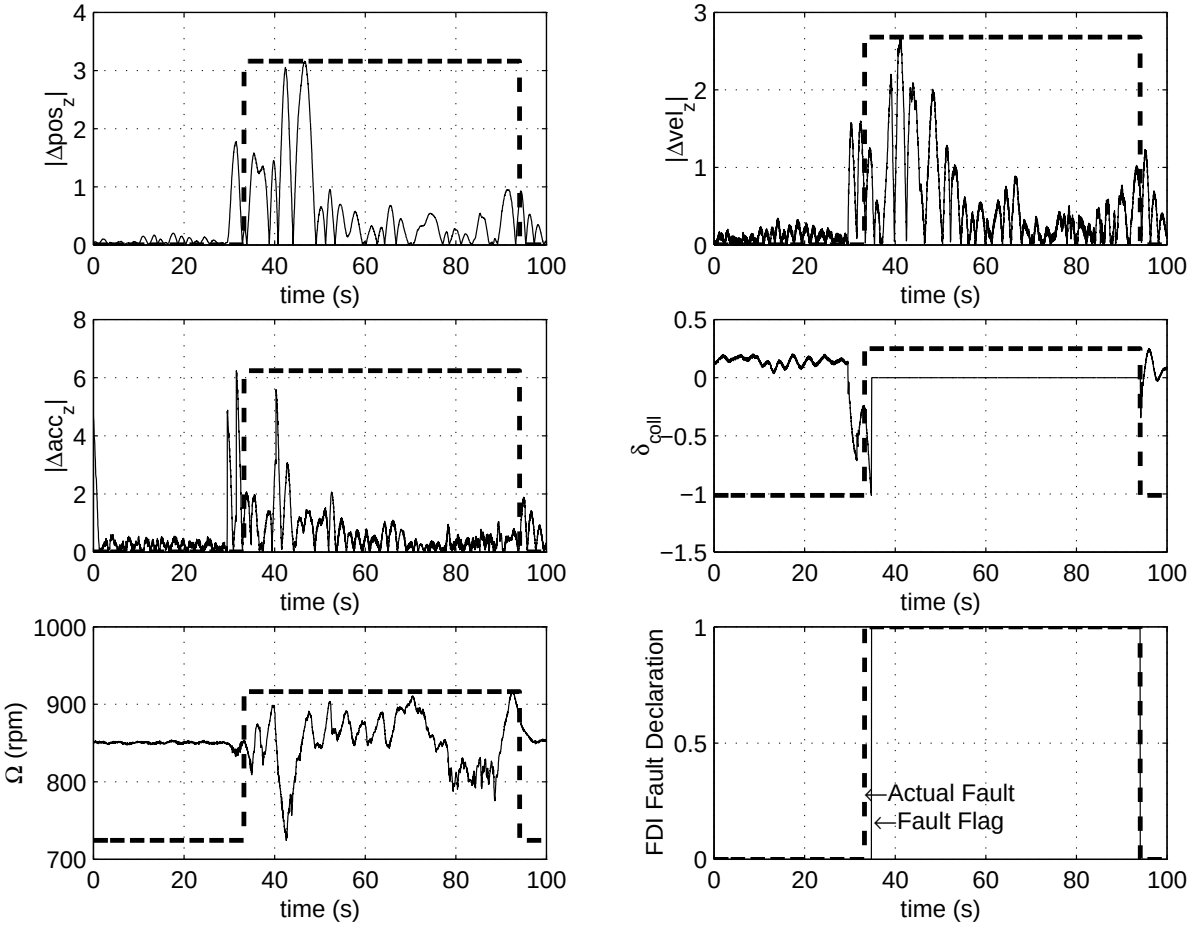

Figure 23. Signal plots for the FDI neural network before and after a collective actuator malfunction on the GTMax. The dashed line indicates the occurrence of the fault. Reference test #30 on Table 2.

trajectories. The aircraft was downwind from 50-70 seconds and upwind from 80-95 seconds. Tracking is noticeably improved when the aircraft is upwind; the main rotor RPM was also lower.

The FDI routine reduced its detection time from over 3 seconds on the previous test to less than 2 seconds. This occurred despite the aggressive flight path because a larger set of training data was available for the neural network when this test was conducted. Aggressive flight paths complicate the FDI process for state-dependent FDI algorithms because intentional aggressive maneuvers can manifest themselves like fault modes. Figure 23 depicts the inputs signals for the collective actuator FDI neural network: vertical position, velocity, and acceleration error magnitudes, the commanded collective pitch and the main rotor speed. The hashed box indicates the presence of the fault. The transients in the signals prior to the onset of the fault were caused by the commanded downward acceleration of the aircraft. Once the neural network was adequately trained, it efficiently discriminated fault modes from other disturbances.

5.2.2 Actuator malfunctions with sensor-dependent fault detection and identification

In another flight test, the FACT FDI routine was employed to detect an immobilized swashplate actuator while the aircraft hovered around a 50-foot square (Figure 24). The helicopter maintained a constant heading, oriented into the wind, through the maneuver. It started the circuit in the lower corner of the figure and proceeded counter-clockwise. The right swashplate actuator was immobilized at the hover trim setting during the first leg of the square (Figure 25). The FDI component issued the correct fault declaration in approximately one second; the active system restructuring component activated the appropriate restructuring and reconfiguration. The baseline flight controller operates at reduced control bandwidths when a swashplate actuator reconfigurable flight controller is active.

The flight test results presented in this chapter validated the use of Table 3 for active system restructuring on the GTMax. They also exposed the strengths and weaknesses of the two approaches for FDI. False positives were significant factor in the development of the FDI neural network, yet a false positive never occurred during the development of

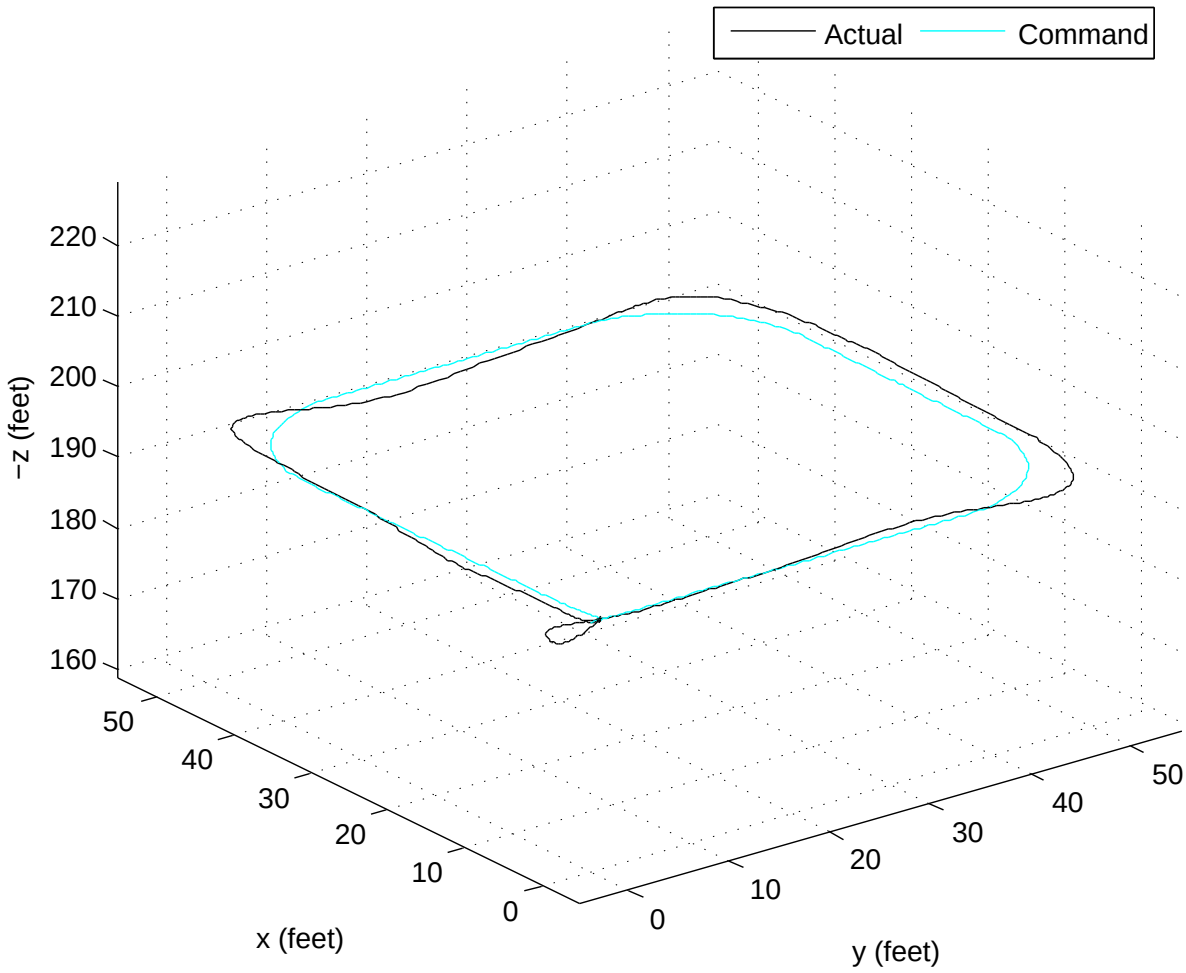

Figure 24. Flight trace for an immobilized right swashplate actuator with integrated FACT FDI on the GTMax. Reference test #28 on Table 2.

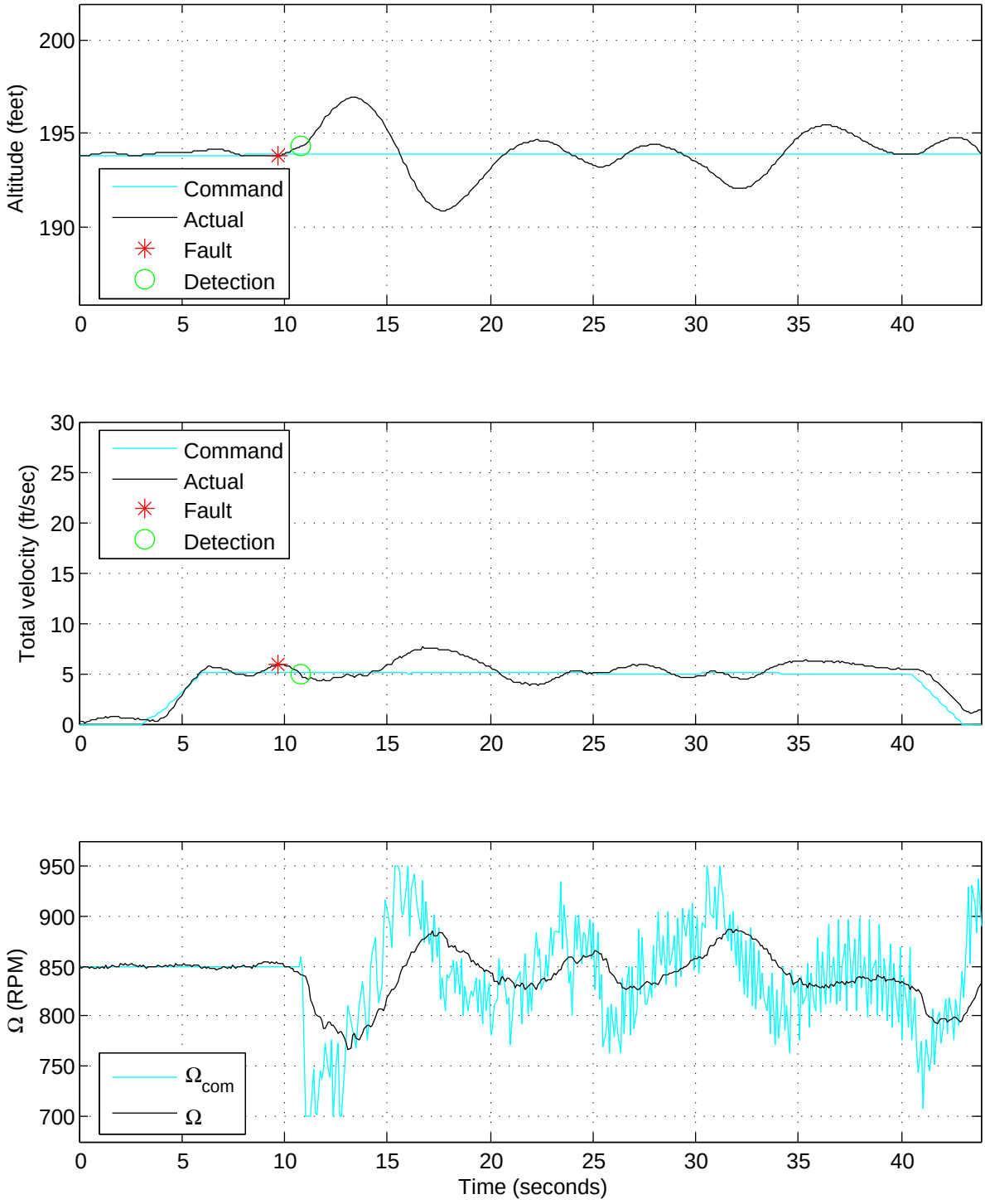

Figure 25. Flight data for an immobilized right swashplate actuator with integrated FACT FDI on the GTMax. Reference test #28 on Table 2.

the FACT FDI routine. The use of simulated sensors on the actuators also afforded the FACT FDI routine a faster detection time. The latency associated with fault detection was not problematic for either FDI routine. This success alludes to another benefit of using an emergency procedure database. Restructuring instructions are not delayed by the numerical optimization of a performance function. On the other hand, such an optimization would provide a better means of accommodating faults that only partially degrade the actuator response (Reference Equation 1, $0 < K_{act} < 1$.). In this case, the optimization of a performance function could evaluate whether the severity of a fault necessitates activating a reconfigurable flight controller. A performance function that utilizes the optimization conducted by reconfigurable path planner (Chapter 6.) could be devised.

CHAPTER 6

RECONFIGURABLE PATH PLANNING

UAV flight paths are typically constructed using a conservative reference model. Use of a conservative model ensures that the resulting flight paths are attainable in adverse environmental conditions. However, even extremely conservative flight paths can exceed the capabilities of an aircraft after the onset of a fault. For this reason, reconfigurable path planning is critical to fault-tolerant control. Figure 26 depicts a flight trace of the GTMax with an immobilized collective actuator during a flight test conducted last year. This flight was conducted on the same afternoon as the flight depicted in Figure 21. In fact, the waypoints and commanded flight path for the two flights were also identical. The baseline path planner generated the flight path in each flight. The difference in the two flights was the nature of the fault that occurred. In Figure 21, the collective was immobilized in a descent setting, a low power setting. In Figure 26, the collective was immobilized at a high power setting. During the later portion of the trace in Figure 26, the reconfigurable flight controller tracked the commanded path quite well despite the elevated collective setting. Of course, at the bottom of the initial descent, where the helicopter was asked to decelerate vertically and horizontally, the reconfigurable flight controller could not adequately track the command trajectory. The goal of the fault-tolerant control architecture is to expand the usable envelope of the fault-impaired unmanned aircraft towards the physical limits of the degraded airframe. In this test, the collective was immobilized at a particularly difficult setting, at a particularly difficult time, and the path planner generated a flight path that exceeded the capability of the impaired system. This example emphasizes the importance of reconfigurable path planning in recovering mission utility from a degraded UAV. The remainder of this chapter develops a reconfigurable path planning algorithm to ensure that the generated flight paths are suitable for the degraded system. The reconfigurable path planning algorithm is an indirect adaptive, receding horizon, model predictive, trajectory

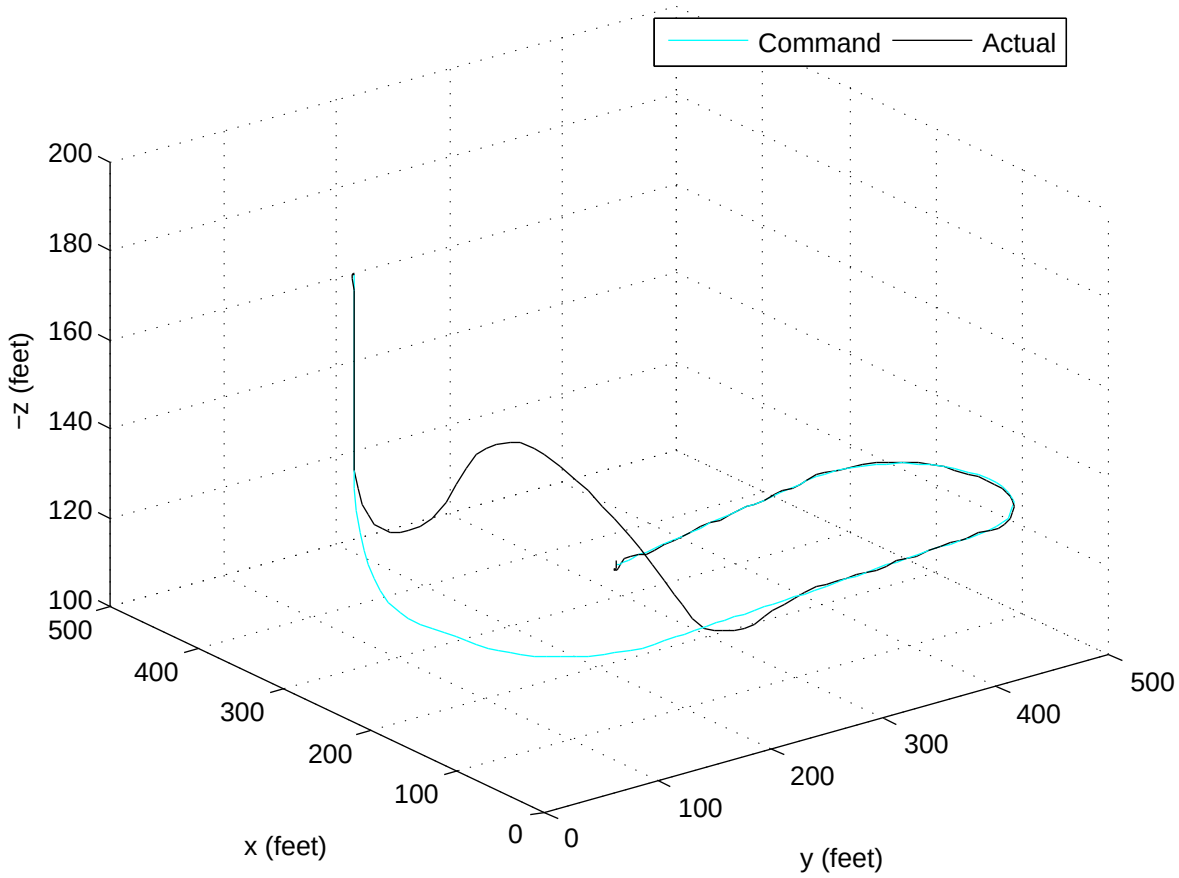

Figure 26. Flight trace with the collective immobilized at a high power setting in forward flight on the GTMax. Reference test #31 on Table 2.

generation routine that runs in real-time.

The reconfigurable path planner component includes an instance of the baseline path planner. It refines the flight paths generated by this copy of the baseline path planner to create its improved flight paths. The two copies of the baseline path planner are identical although the mission adaptation component discussed in the next chapter can modify certain parameters that change the trajectories generated by the second copy of the baseline path planner. The two copies of the baseline path planner also differ in operation. The second copy operates at a lower frequency and several seconds in front of the vehicle, on the receding horizon. The primary baseline path planner operates at the same frequency as the baseline flight controller with no receding horizon. The baseline path planners on the GTMax and the Renegade UAV both integrate step functions in jerk to generate flight paths. This problem can be reduced to a minimum-time constrained linear optimal control problem for which the solution is a bang-coast-bang type control. Solving the problem off-line generates switching conditions for the planner. Here are the steps of the baseline process:

1. Determine the direction of the required jerk command based on the switching conditions.

2. Integrate to obtain the acceleration command. The switching conditions ensure that the acceleration command does not exceed a chosen maximum nominal acceleration.

3. Integrate to obtain the velocity command. The switching conditions ensure that the velocity command does not exceed a chosen target velocity.

4. Integrate to obtain the position command.

The process is applied in three dimensions. Three parameters, namely the saturation levels for jerk, acceleration, and velocity, characterize the flight paths generated by this method.

6.1 Linear Programming for Reconfigurable Path Planning

The reconfigurable path planning component takes two inputs to generate an optimized flight path. It receives a discrete-time linear model of the vehicle dynamics from the system

identification component,

$$x(k+1) = Fx(k) + Gu(k) + d(k)$$
$$y(k) = Cx(k). \tag{24}$$

It also receives waypoints from the third tier of the control hierarchy (Figure 5). Prior to the optimization process, the commanded flight path is generated from the waypoints using the algorithm discussed above. This flight path is maintained as a queue of commanded positions; it is updated and shifted every sample time. Whenever a waypoint is added, deleted, or changed, the entire path must be recalculated. The reconfigurable path planner uses the approximate linear model of the vehicle dynamics to reshape the commanded flight path of the vehicle. The constrained optimization problem is posed as a linear program [23]. Posing the optimization as a linear program has four advantages:

- Using modern software, optimization over a useful window is feasible in real-time.
- Bounds on the actuator positions, actuator rates, state variables, and output errors are easily applied.
- Optimization can be formatted to minimize the 1-norm, the ∞-norm or a combination of the two.
- The discrete linear model of the system can be changed at any time step in the optimization window.

Generally, one cannot assume that a UAV will maintain linear dynamics across an optimization window that lasts several seconds. The inaccuracy that results from this assumption is mitigated by two features of the chosen design. First, only the first step of the reconfigurable flight path is sent to the controller. Presumably, the linear model is most accurate for the first time step. State-dependent Riccati equation control employs a similar assumption [9]. Second, if predictable changes to the linear model are expected during the optimization window, approximations for those changes can be applied during the formation of the linear program. Of course, linear programming excludes the state-dependence of the linear model from the optimization process.

To approximate the system response over a window of N sampling periods, the vector $\bar{x}$ is formed by concatenating $x(k)$ through $x(k+N-1)$ from Equation 24. The following matrices are formed:

$$\bar{F} = \begin{bmatrix} F \\ F^2 \\ : \\ F^N \end{bmatrix}$$

$$\bar{G} = \begin{bmatrix} G & 0 & .. & 0 \\ FG & G & .. & 0 \\ : & : & : & : \\ F^{N-1}G & F^{N-2}G & .. & G \end{bmatrix}$$

$$\bar{d} = \begin{bmatrix} d \\ Fd + d \\ : \\ \sum_{i=1}^{N} F^{i-1}d \end{bmatrix}$$

To complete the system $\bar{u}$, $\bar{y}$, and $\bar{C}$ are also formed accordingly,

$$\begin{aligned} \bar{x}(k+1) &= \bar{F}x(k) + \bar{G}\bar{u}(k) + \bar{d}(k) \\ \bar{y}(k) &= \bar{C}\bar{x}(k). \end{aligned} \tag{25}$$

A suitable goal for the optimization process is to minimize the following objective function:

$$J(\bar{u}) = f^T|\bar{y} - \bar{y}_{com}| \tag{26}$$

subject to state and control saturations. $\bar{y}_{com}$ is the flight path that is generated from the waypoints expressed in the body frame. f is simply a weighting vector for the error vector, $|y - y_{com}|$. An error bound vector Δ is introduced to resolve the absolute value,

$$-\Delta \leq \bar{y} - \bar{y}_{com} \leq \Delta. \tag{27}$$

To cast the problem in the standard form for a linear program,

$$\begin{aligned} \min\ & \tilde{c}^T \tilde{x} \\ \text{subject to } & \tilde{A}\tilde{x} \leq \tilde{b} \\ & \tilde{l} \leq \tilde{x} \leq \tilde{u} \end{aligned} \tag{28}$$

where the ˜ notation is introduced so that elements from the linear program are not confused with control variables. The elements of the linear program are formed as follows:

$$\tilde{x}^T = \begin{bmatrix} \bar{u} & \Delta \end{bmatrix}$$

$$\tilde{A} = \begin{bmatrix} \bar{C}\bar{G} & -I \\ -\bar{C}\bar{G} & -I \end{bmatrix}$$

$$\tilde{b} = \begin{bmatrix} -\bar{C}(\bar{F}x(k) + \bar{d}) + \bar{y}_c \\ \bar{C}(\bar{F}x(k) + \bar{d}) - \bar{y}_c \end{bmatrix}$$

$$\tilde{c}^T = \begin{bmatrix} 0..0 & f \end{bmatrix}$$

where I is the identity matrix. The vectors $\tilde{l}$ and $\tilde{u}$ are formed to include the bounds on the error, Δ and/or $\bar{u}$ although applying overly restrictive bounds to the error Δ can result in an infeasible system. The bounds on $\bar{u}$ may be pulled from Equation 1 or set conservatively between δ_{min} and δ_{max}. In a final step, the optimal path, $\bar{y}$ may be determined from $\bar{u}$. This is easily achieved using Equation 25. The first element of the optimal path, $y(k+1)$ is sent to the baseline flight controller and the reconfigurable flight controllers as necessary. The controls, $\bar{u}$, that are generated during the optimization process are not used for low-level control. In most instances, the entire path $\bar{y}$ is not required. In this case, $y(k+1)$ can be found directly from Equation 24 pulling $u(k)$ from $\bar{u}$.

Referring to back to Equation 24, the dimension of x is $n \times 1$; the dimension of u is $m \times 1$; and the dimension of y is $o \times 1$. Allocating two constraints per actuator and no bounds on the error, $\bar{y} - \bar{y}_{com}$, the linear program in Equation 28 has $N(m+o)$ variables and $2No$ constraints.

The linear program in Equation 28 minimizes the $1-$norm of the error, $\bar{y} - \bar{y}_{com}$. The problem can easily be posed to minimize the $\infty-$norm. Doing so reduces Δ to a scalar significantly reducing the number of unknowns in the linear program. The number of variables is reduced, but the number of constraints remains unchanged. Nonetheless, a reduction in computation time may result. A disadvantage of the $\infty-$norm is that the optimized path often includes oscillations with the magnitude of the error bounded by the $\infty-$norm. An alternative to the $\infty-$norm and the $1-$norm is to divide the problem into multiple sections and optimize each section using the $\infty-$norm. As the number of sections approaches N, the $1-$norm is achieved.

6.2 Implementation with System Identification Based Adaptation

The system identification component takes in the vehicle's state and its fault condition, and it constructs a discrete linear model of the vehicle dynamics. The low-level controllers do not utilize this model. This reduces interaction between the system identification process and the vehicle dynamics which can arise in indirect adaptive control schemes. The process estimates a continuous-time model of the system at a relatively high frequency, 25 Hz on the GTMax, and converts the continuous-time model to a discrete model when necessary for the optimization process, at 2.5 Hz on the GTMax. Using an extended sampling period in the optimization process allows extra time for computation without degrading precision. The control bandwidths of the closed loop system are significantly slower than 2.5 Hz $= 5\pi$ radians per second.

The results presented in this chapter employed a simple adaptation to improve the accuracy of the vehicle model across a variety of flight profiles. Matrices F and G (Equation 24) were pre-determined based on the content of the fault declaration issued by the FDI component. The following adaptation was applied to the vector d to account for model inaccuracies:

$$d(k+1) = d(k) + KTe(k) \tag{29}$$

where K is the adaptation gain. An expression for the error function, $e(k)$, is provided in

the next section. While a number of more complex methods for system identification are available, this simple implementation was effective and enhanced the performance of the reconfigurable path planner across a wide range of airspeeds. Moreover, excluding F and G from adaptation reduced computation time. The matrices $\bar{F}$, $\bar{G}$, as well as the products, $\bar{C}\bar{F}$ and $\bar{C}\bar{G}$, were constant and therefore computed offline. $\bar{d}$, $\bar{C}\bar{d}$, and $\bar{y}_{com}$ were computed online.

6.3 GTMax Simulation Results

The fault-tolerant control architecture including the reconfigurable path planner was applied to a non-linear simulation of the GTMax [23]. The simulation was distributed on two personal computers. Splitting the computation is routine as the actual aircraft includes two onboard computers. The first computer ran the baseline flight controller, the baseline path planner, active system restructuring, and the mission assignment components as well as the simulation environment. Algorithms on this computer operated at 50 Hz. The second computer ran the system identification, reconfigurable path planning, and reconfigurable flight controller components. The reconfigurable flight controller and the system identification component ran at 25 Hz while the reconfigurable path planner ran at 2.5 Hz, $T = 0.4$ seconds. The linear program was assembled to minimize the $1-$norm over a 12-second window, $N = 30$. Optimization was conducted on the second computer using ILOG CPLEX 9.0. The linear program consisted of 150 variables and 120 constraints; CPLEX used the simplex method to solve the program. If the optimization process failed to terminate within $T = 0.4$ seconds, the path planner integrated the previous solution forward and continued to work the solution. However, this rarely occurred.

The following continuous-time model of the helicopter's longitudinal dynamics was discretized for use in the optimization process:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{30}$$

where $x = [u, w, q, x, z, \theta, \Omega]^T$ and $u = [\delta_{coll}, \delta_{lon}, \Omega_c]^T$. A and B were approximated as

follows:

$$A = \begin{bmatrix} X_u & 0 & 0 & 0 & 0 & -g & 0 \\ 0 & Z_w & Z_q & 0 & 0 & Z_\theta & Z_\Omega \\ M_u & M_w & M_q & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1/\tau \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ Z_{\delta_{coll}} & 0 & 0 \\ 0 & Z_{\delta_{lon}} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/\tau \end{bmatrix}$$

C was chosen so that $y = [x, z]^T$. f^T was chosen as [1 10 .. 1 10]. The longitudinal error Δ_x was bounded at 100 feet. This selection for the bound never resulted in an infeasible linear program. If an infeasible program had occurred, the reconfigurable path planner would have reformed the linear program without bounding the error. Mission adaptation discussed in the next chapter offers an alternative to removing the error bounds. During the optimization, control inputs were allowed half their full range of motion. Ω_{com} was bounded between 725 and 925 revolutions per minute; the nominal rotor speed is 850 revolutions per minute.

To demonstrate the utility of reconfigurable path planning in conjunction with adaptive neural network reconfigurable flight control, the helicopter was commanded to execute a sequence of maneuvers that was known to result in an unacceptable baseline flight path. The collective actuator was immobilized near hover trim for all simulations, and the aircraft was under active RPM control throughout. The sequence of maneuvers was executed three

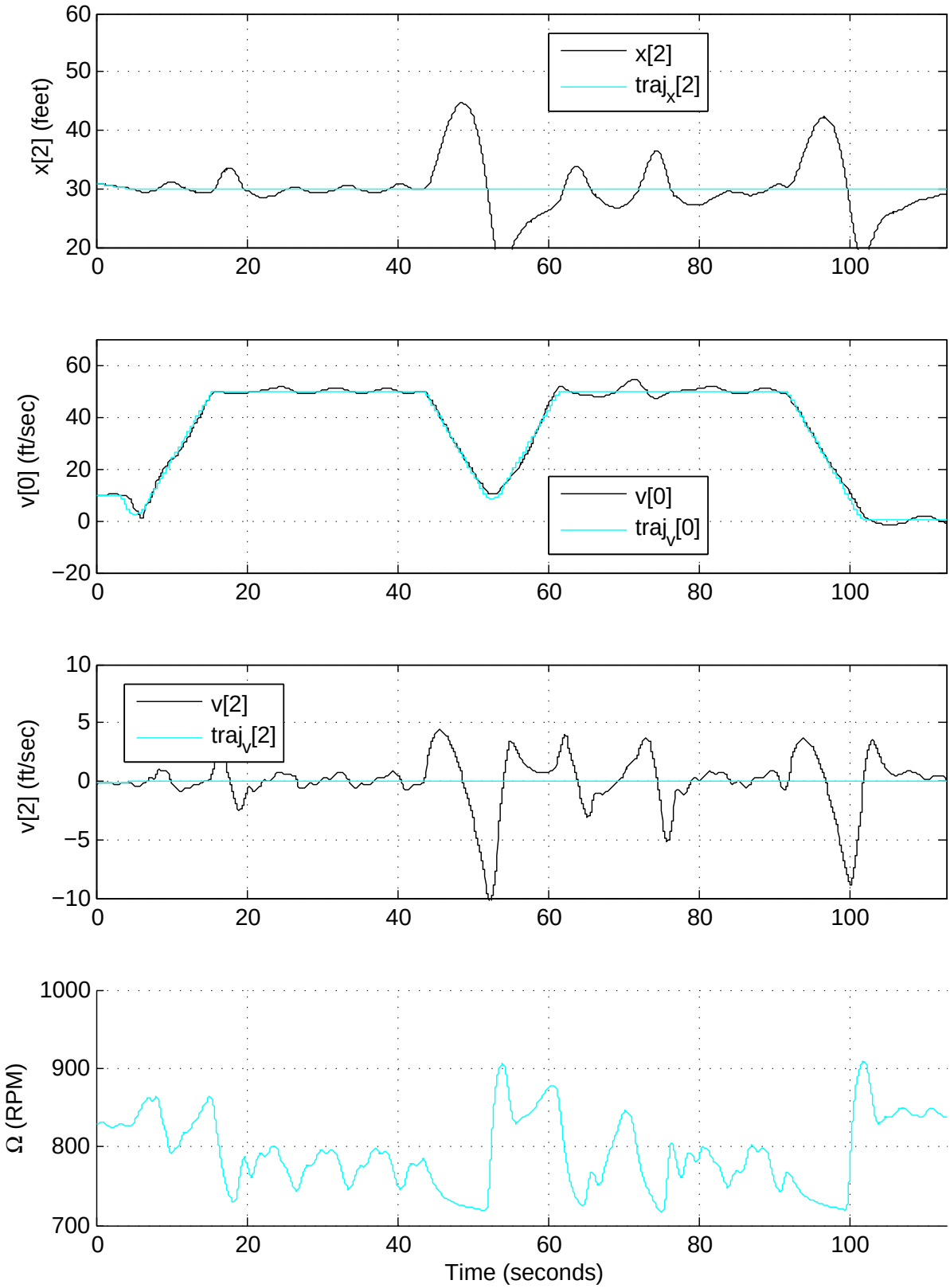

Figure 27. Simulation data with the collective actuator immobilized using a PID reconfigurable flight controller and the baseline path planner on the GTMax.

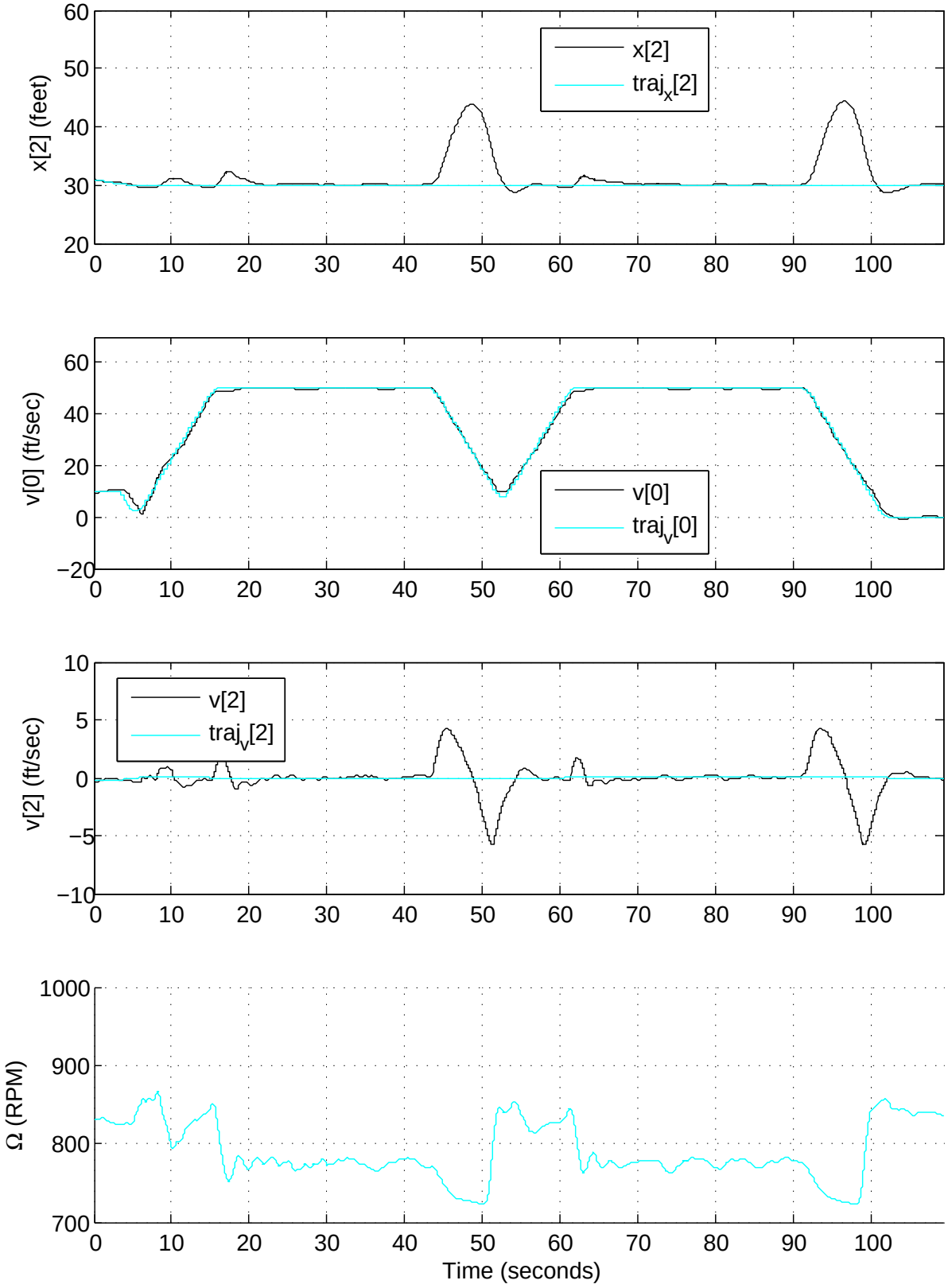

Figure 28. Simulation data with the collective actuator immobilized using an adaptive neural network reconfigurable flight controller and the baseline path planner on the GTMax.

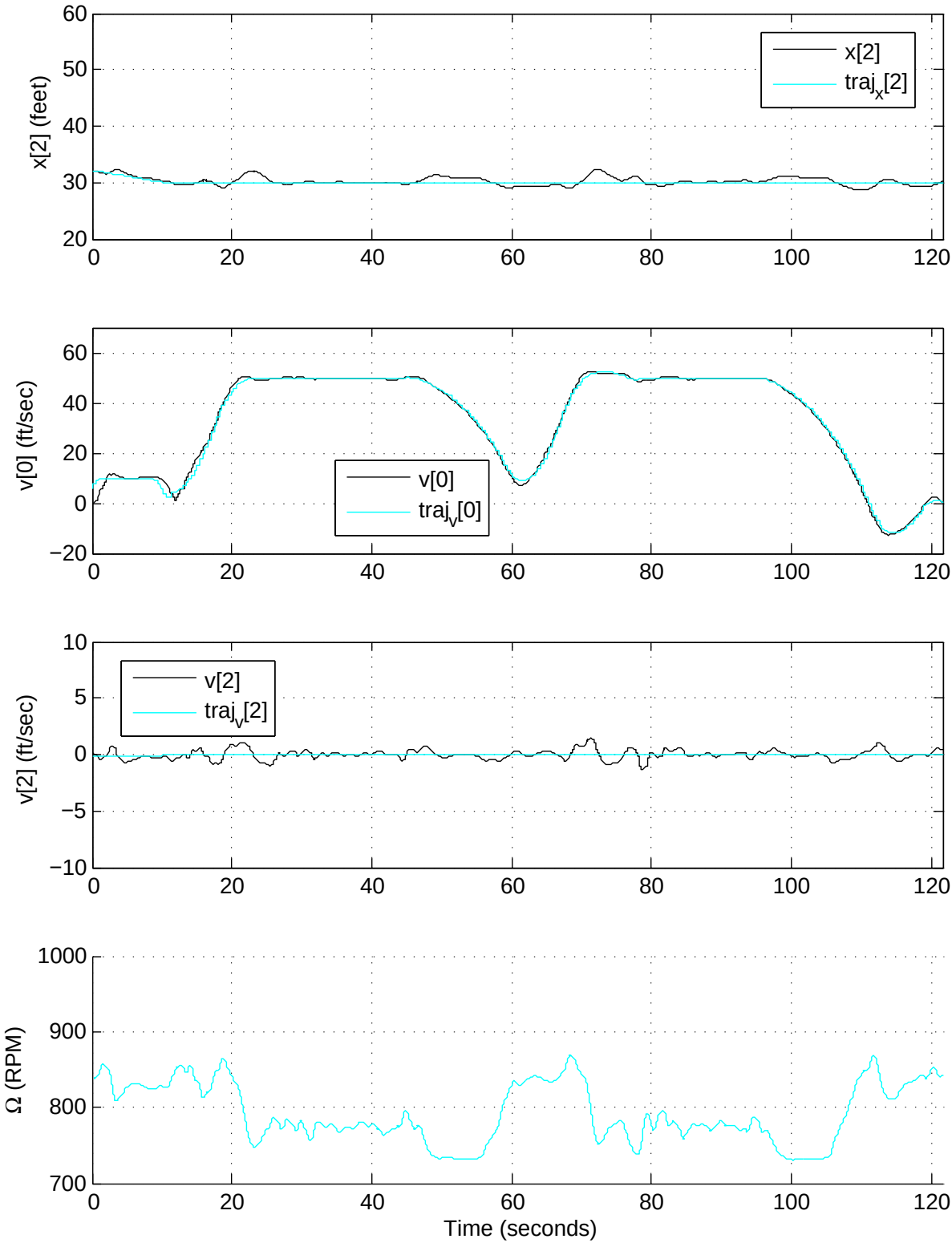

Figure 29. Simulation data with the collective actuator immobilized using an adaptive neural network reconfigurable flight controller and the reconfigurable path planner on the GTMax.

times: once using the baseline path planner with a PID RPM controller (Figure 27), once using the baseline path planner with an adaptive neural network RPM controller (Figure 28), and once using the reconfigurable path planner with an adaptive neural network RPM controller (Figure 29). The helicopter starts the sequence moving forward at 10 feet per second. Subsequent waypoints direct the helicopter to move forward at [50, 0, 50, 0] feet per second. The assigned altitude is 30 feet throughout the sequence. As mentioned earlier, the baseline path planner applies a saturation function to accelerations in its flight paths. The 5 feet per second per second saturation is clearly visible in Figures 27 and 28. The adaptive neural network flight controller provides a significant improvement over the PID controller, but it still cannot maintain the helicopter's altitude when the helicopter pitches up to decelerate. In both cases using the baseline path planner, the altitude error nears 15 feet. By smoothing the velocity trajectory, the reconfigurable path planner held the altitude error during both decelerations to less than 3 feet (Figure 29). Notice that Ω is held at its lower bound for a large portion of both deceleration arcs. The reconfigurable path planner decided to arrive late at both stopping points in order to maintain altitude.

The results in this chapter validate the use of reconfigurable path planning as a means to augment fault-tolerant control. The simulation results clearly indicate that improved aircraft path planning and aircraft performance can be achieved using linear programming. Reconfigurable path planning allows the vehicle to assess its condition and generate flight paths that closely match the capability of the aircraft. Extending reconfigurable path planning to operate before the occurrence of a fault can improve the performance of the nominal system.

CHAPTER 7

MISSION ADAPTATION

Given a sequence of waypoints, the reconfigurable path planning component optimizes the flight path of the aircraft. However, reconfigurable path planning cannot generate an acceptable flight trajectory unless the sequence of waypoints assigned by the third tier of the control hierarchy is within the capability of the aircraft. The mission adaptation component evaluates the capability of the vehicle and constrains the assigned waypoints depending on the severity of the fault mode. This chapter develops two methods to evaluate the aircraft capability. One is based on the optimization process conducted by the reconfigurable path planner; a second relies directly on system identification. Both methods quantify the capability of the closed loop system in parameters such as maximum acceleration or maximum velocity that are easily exportable to higher level planning algorithms. In general, higher level planning algorithms require a simple closed loop model of every vehicle involved in the mission planning sequence. The use of simple models reduces the complexity of the mission planning problem to an acceptable level so that techniques such as stochastic differential games are useful [62]. The DARPA HURT program, which recently completed Phase I flight demonstrations, employs simplified vehicle models to coordinate the operation of multiple UAVs conducting a collective surveillance task.

Chapter 2 presented the objective of the entire fault-tolerant control architecture using the following equation:

$$J(M, R) = U(Pe, M, M_{com}).$$

The subsequent chapters developed reconfigurable flight controllers, active system restructuring, and reconfigurable path planning for optimizing the performance of the aircraft, Pe, through restructuring and reconfiguration, R. Each of these functionalities differs distinctly from mission adaptation, which occurs at the third tier of the fault-tolerant control architecture (Figure 5). The first and second tier functions affect the usefulness of the UAV, U, in

accomplishing its assigned mission, M_{com}, by improving vehicle performance, Pe. Mission adaptation allows the control architecture to pursue relaxed mission objectives M in order to achieve greater vehicle usefulness. Both the M and M_{com} are expressed as a sequence of waypoints. Mission adaptation alters parameters of the individual waypoints, such as the nominal jerks, accelerations and velocities used by the path planners to generate flight paths. Alternate implementations of mission adaptation could move the actual location of waypoints or prune waypoints from the sequence. At a minimum, mission adaptation implies a change in the aircraft's time of arrival at one or more waypoints. By adapting select waypoint parameters, the mission adaptation component enables the aircraft to accomplish an altered mission with increased usefulness.

As mentioned in the previous chapter, the reconfigurable path planner contains a copy of the baseline path planner. This copy generates a flight path that becomes the optimization goal of the reconfigurable path planner. Therefore, if the baseline flight path grossly exceeds the capability of the degraded aircraft, the reconfigurable path planner cannot compensate. In the case where the linear program includes bounds on the error vector Δ, an overly aggressive baseline flight path can result in an infeasible linear program. For instance, if the baseline path planner commands the aircraft to exceed its maximum velocity, the position error will increase in time, and the linear program cannot remain feasible. This situation is alleviated by adapting the baseline path. Successful implementation of mission adaptation serves to protect the reconfigurable path planning component from infeasible flight paths.

When constructing flight paths, the baseline path planner integrates a set of parameters that are assigned for each waypoint. These parameters such as the nominal jerk, acceleration, and target velocity are assigned conservatively so that they do not exceed the capability of the baseline flight controller. After the occurrence of a fault, the mission adaptation component implements an appropriate adjustment to the parameters. Changes to the parameters can be implemented within the receding window employed by the reconfigurable path planner.

7.1 Optimization Based Mission Adaptation Applied to the GTMax

For decades, linear programming has been employed to generate minimum time solutions to discrete-time optimal control problems [84]. Under a typical process, an optimal control solution is generated for a fixed final time. If a feasible solution is achieved, the process is repeated with a reduced final time. Conversely, if a feasible solution is not achieved, the process is repeated with an increased final time. In either case, the process is repeated until the feasible solution with the minimum final time is located. Optimization based mission adaptation follows a similar process, but it affects the final time indirectly by changing waypoint parameters such as the nominal jerk and nominal acceleration used by the baseline path planner.

The reconfigurable path planner developed in the previous chapter minimizes the following function:

$$J(\bar{u}) = f^T|\bar{y} - \bar{y}_{com}|. \tag{31}$$

Optimization based mission adaptation adjusts waypoint parameters to bound $J(\bar{u})$ using the following algorithm:

- Conduct the reconfigurable path planning optimization.
- If $J(\bar{u}) > kN$, adjust the waypoint parameters.

Again, N is the length of the optimization window and k is a scalar weight. Essentially, it's the maximum acceptable mean error value. The process identifies the closed loop performance parameters that allow the aircraft to perform acceptably. The control designer has the freedom to replace the condition, $J(\bar{u}) > kN$, as necessary to achieve his objective. However, the method used to adjust the waypoint parameters must be selected carefully. For instance, decrementing a waypoint target velocity can have a detrimental affect. Decreasing the nominal acceleration and/or jerk generally results in more attainable flight paths.

7.1.1 Simulation results

To demonstrate the mission adaptation component, the GTMax was commanded to execute the same sequence of waypoints used in the previous chapter. However, for this demonstration the collective was immobilized at a higher setting than in the previous chapter. As a result, the restructured aircraft is limited in its ability to accelerate downward. For the first demonstration, all components of the fault-tolerant control architecture excluding mission adaptation were active. Despite the use of reconfigurable path planning, the aircraft could not maintain its altitude during deceleration (Figure 30).

Figure 31 depicts the aircraft response using optimization based mission adaptation. During the simulation, the mission adaptation component, decrements the nominal acceleration and nominal jerk of the baseline path planner if $J(\bar{u}) > 10N$. During the initial portion of the flight the aircraft reduces the nominal jerk and acceleration settings to match the capability of the aircraft. By $t = 53$ seconds, the mission adaptation component settles on its final values for the nominal jerk and nominal acceleration. Notice that the aircraft does not attempt to stop at the middle waypoint. Once the mission adaptation component opts to reduce its nominal acceleration, it is too late for the aircraft to successfully decelerate using the reduced value. If overshooting, a waypoint is not an acceptable consequence, the mission adaptation component could be revised so that it operates sufficiently far in front of the aircraft. As expected, mission adaptation delays the aircraft's arrival at its third stop.

7.2 System Identification Based Mission Adaptation Applied to the Renegade UAV

Mission adaptation on the Renegade UAV uses a system identification based approach. The mission adaptation component processes information from the system identification component to impose limits on closed loop path planning parameters: nominal acceleration and nominal jerk. The system identification component estimates maximum feasible values for these parameters using a standard linear model of the vehicle which can be developed through system identification. The A and B matrices for the Renegade UAV give the

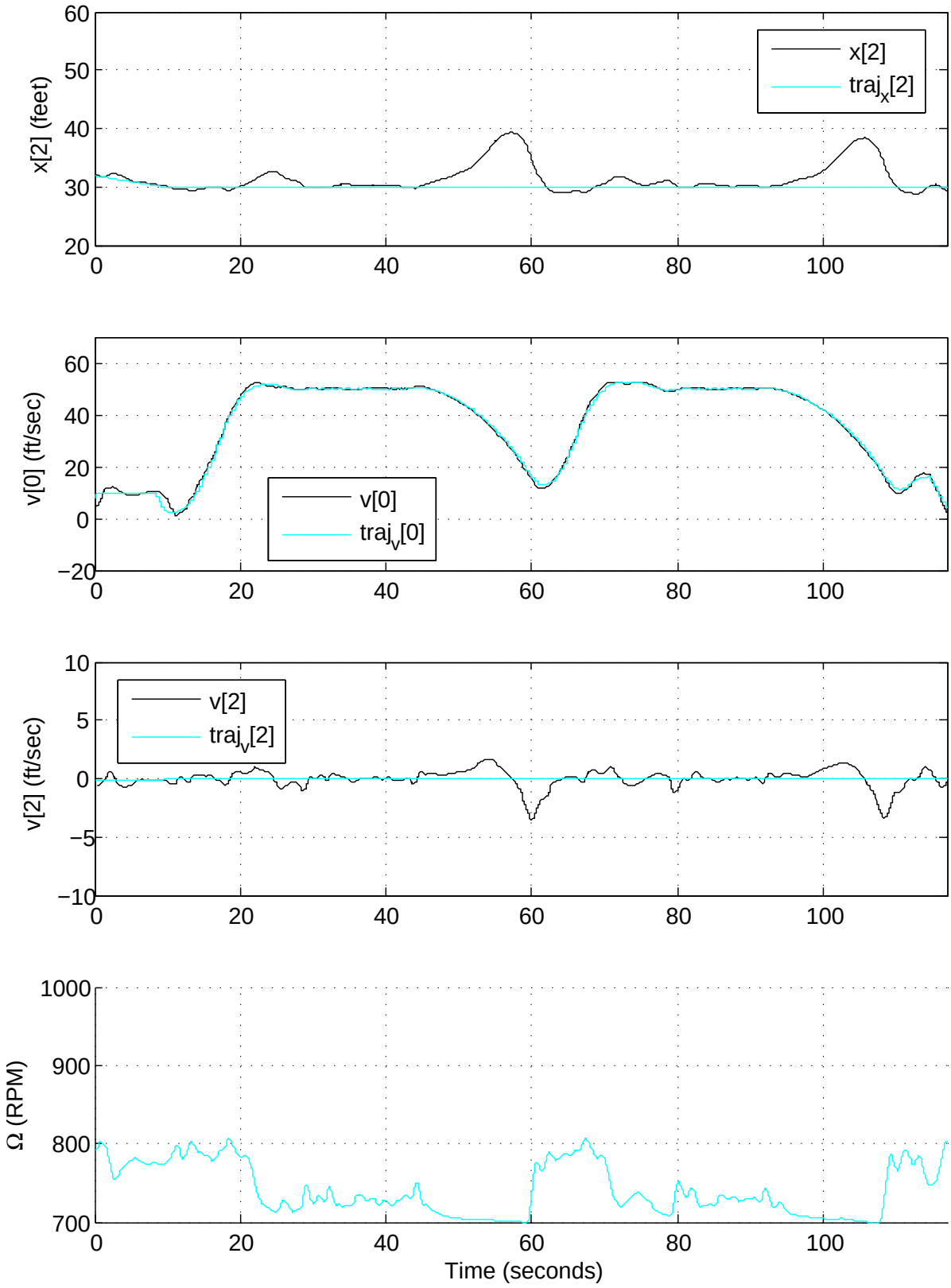

Figure 30. Simulation data with the collective actuator immobilized at a high setting with reconfigurable path planning and without mission adaptation on the GTMax.

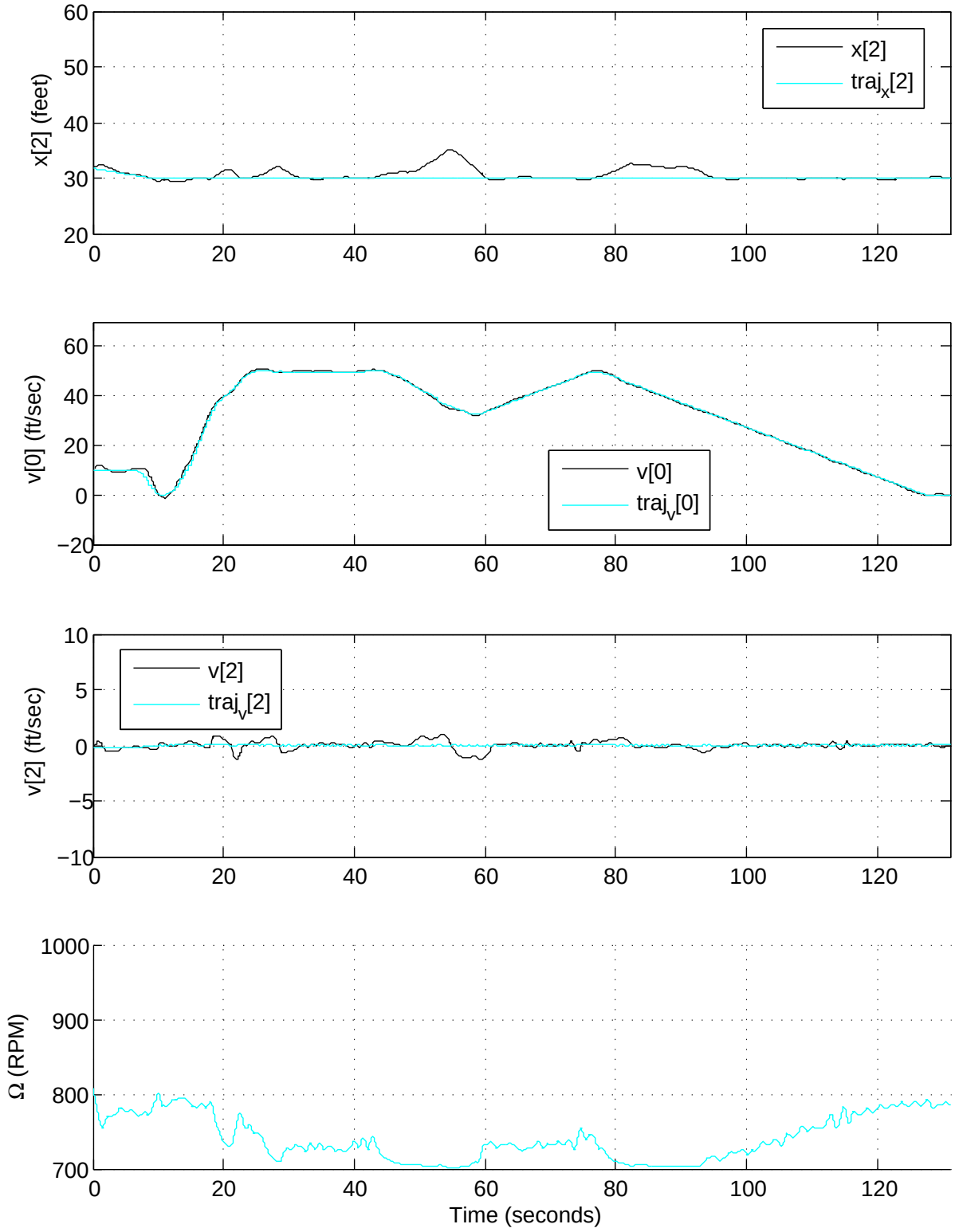

Figure 31. Simulation data with the collective actuator immobilized at a high setting with reconfigurable path planning and mission adaptation on the GTMax.

following:

$$\dot{u} = X_u u - g\theta$$

$$\dot{w} = Z_w w + Z_q q + Z_\theta \theta + Z_{\delta_f} \delta_f.$$

The estimate for $\dot{u}_{max}$ is based on the assumption that in steady-state, $q \approx w \approx \dot{w} \approx 0$ and all remaining collective, $\delta_{max} - \delta_f$, can be converted to increase forward acceleration:

$$\dot{u}_{max} = \frac{g Z_{\delta_f}}{Z_\theta}(\delta_{max} - \delta_f) + \dot{u}. \tag{32}$$

A parallel estimate for $\dot{u}_{min}$ is achieved using δ_{min}. Applying a low pass filter to these estimates generates consistent values for $\dot{u}_{max}$ and $\dot{u}_{min}$. De-activating the adaptation of $\dot{u}_{max}$ and $\dot{u}_{min}$ during certain aggressive maneuvers enhances the identification process. The mission adaptation component takes the value of smaller magnitude and rounds it to the nearest 0.5. The rounded value becomes the nominal acceleration used in path planning. This method of determining the nominal acceleration ignores limitations on vertical acceleration, $\dot{w}$, but vertical accelerations are generally much shorter in duration.

7.2.1 Simulation results

Boeing's test facility for the Renegade UAV is located in the California high dessert, Victorville, California. During summer months, the high altitude and temperature creates adverse operating conditions for the under-powered Renegade UAV. The following simulations place the aircraft at 4000 feet above mean sea level, at 95 degrees Fahrenheit. Under these conditions, the Renegade UAV can approach collective saturation, and its engine is easily over-strained. Like most UAVs, the Renegade UAV does not adjust its path planning mechanism based on the ambient temperature or pressure altitude. The aircraft is commanded to accelerate forward to 101 feet/second (60 knots), cover a specified distance, and then decelerate to a stationary hover. Figure 32 depicts the response of the aircraft without mission adaptation online. The flight includes large altitude errors, collective inputs which approach saturation at both ends of the scale, and most significantly a main rotor RPM over-speed during the deceleration. Figure 33 depicts the performance of the aircraft with mission adaptation online. The altitude error and collective inputs are both well bounded,

and the main rotor over-speed is avoided. In this demonstration, mission adaptation enables the aircraft to adapt to uncertain environmental conditions.

System identification based mission adaptation is simpler to implement and far less expensive computationally than optimization based mission adaptation. However, its accuracy has a strong dependency on the current state of the vehicle, and it fails to consider the path in front of the aircraft. Optimization based mission adaptation looks forward in time over a receding horizon, and it includes only an indirect dependency on the state of the vehicle. The system identification component, which generates the linear model of the vehicle used in the optimization process, depends on the state of the aircraft. Both methods for mission adaptation employ an incremental search which accelerates the estimation process and prevents the parameters from drifting unnecessarily. Mission adaptation can be employed with or without reconfigurable path planning. The fault-tolerant control architecture employed on the Renegade UAV did not use reconfigurable path planning. When mission adaptation is implemented with reconfigurable path planning, the computational burden associated with optimization based mission adaptation is greatly reduced. In this case, use of optimization based mission adaptation is easily justified.

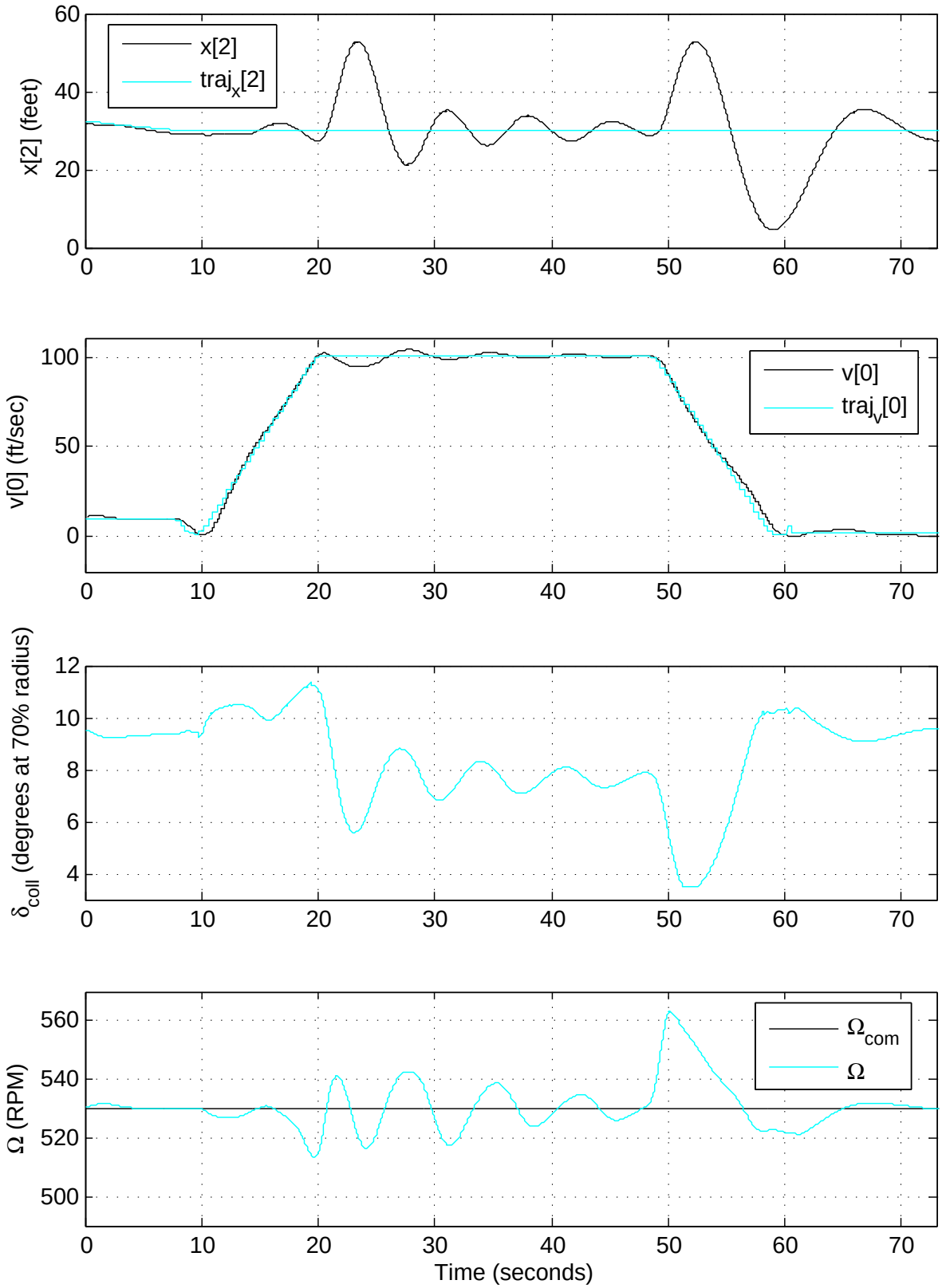

Figure 32. Simulation data at 4000 feet above mean sea level and 95 degrees Fahrenheit without mission adaptation on the Renegade UAV.

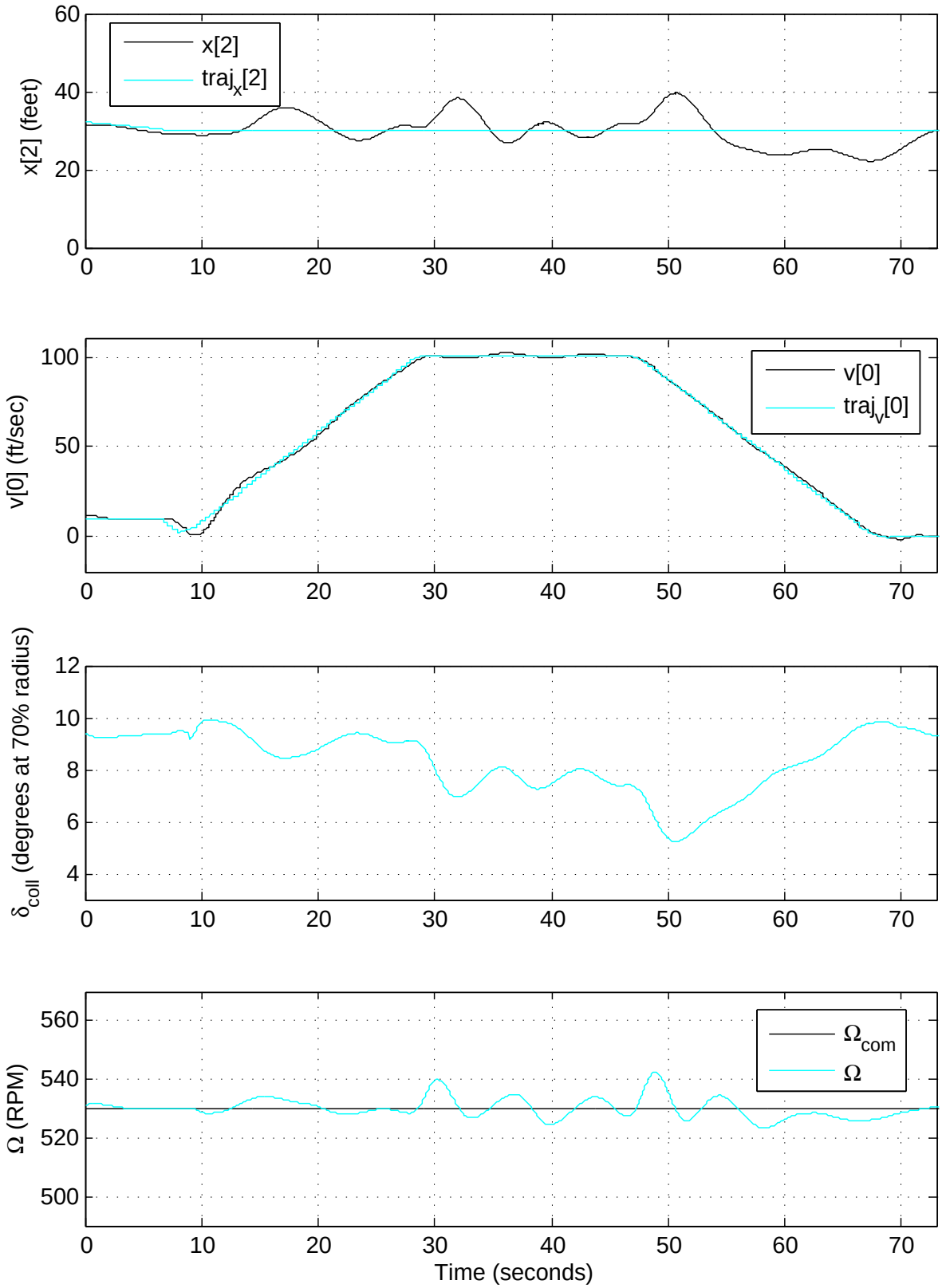

Figure 33. Simulation data at 4000 feet above mean sea level and 95 degrees Fahrenheit with mission adaptation on the Renegade UAV.

CHAPTER 8

CONCLUSIONS AND RECOMMENDED FURTHER RESEARCH

8.1 Conclusions

The fault-tolerant control architecture developed in this research extends the state of the art in fault-tolerant control of UAVs. It addresses two shortcomings that Clements acknowledged in his dissertation: it includes a dynamic set point controller (reconfigurable path planning), and it augments the baseline control architecture with local controllers designed specifically to combat fault modes. The architecture advocates the use of prior knowledge and adaptive flight control to optimize the performance of the aircraft after the occurrence of a fault. Table 4 compares the performance of various algorithms for fault-tolerant control developed in this research using a numerical performance assessment. For each entry on the table, the GTMax was commanded to execute an identical set of waypoints, namely the same waypoints used in the simulations depicted in Figures 28, 29, 30, 31, and 27. The error terms depicted in the table compare the vehicle's actual flight trace to the flight path generated by the baseline path planner, not the reconfigurable path planner. This selection for the error terms quantifies the performance of the reconfigurable flight controller and the reconfigurable path planner simultaneously; it assesses the performance of entire fault-tolerant architecture. The weighted mean square error (MSE) in the table is calculated as follows:

$$E = \frac{1}{N}\sum_{N}[(\frac{1}{10}(x - x_{rm}))^2 + (z - z_{rm})^2] \tag{33}$$

This error formulation corresponds with the ten to one weighting present in the reconfigurable path planning optimization process. The table excludes mission adaptation results because mission adaptation invalidates the performance measure by changing the baseline flight path.

Table 4. Performance comparison for the fault-tolerant control architecture.

Row #	Architecture Tested	Fault Applied	Weighted MSE (feet)	Max Vertical Error (feet)
1	Baseline flight controller, baseline path planner		0.8967	2.2124
2	PID reconfigurable flight controller, baseline path planner (Figure 27)	X	3.7557	14.7067
3	Adaptive neural network reconfigurable flight controller, baseline path planner (Figure 28)	X	2.7863	13.8698
4	Adaptive neural network reconfigurable flight controller and reconfigurable path planner (Figure 29)	X	1.4321	2.1217

Previous fault-tolerant control architectures typically fall into two categories. The first category employs a single adaptive flight controller designed to accommodate a variety of fault modes. These controllers use the same adaptive control law before and after the occurrence of a fault so FDI is not required. On the other hand, without FDI, this approach to fault-tolerance ignores the fault-condition of the vehicle when constructing its control vector, and it cannot restructure the system. The controller used in the first row of Table 4 falls in this category. It performs well in the absence of the fault, but a fault such as the stuck collective actuator, which is applied on the remaining rows of the table, would certainly overcome its capability resulting in loss of control of the aircraft. A second category of fault-tolerant controllers creates a set of individually non-adaptive controllers and switches between them based the output of an FDI routine. The methodology benefits from the information acquired during FDI. However, because the individual controllers are not adaptive, the performance of the controllers suffers from modeling uncertainties and other disturbances. The second row in Table 4 depicts the performance of a PID RPM controller (Figure 27). The architecture developed in this research includes multiple individually adaptive controllers so a precise model of the post-fault dynamics is not required. It applies active system restructuring based on prior knowledge about the structure of the system. The methodology allows the architecture to include a specific response to a number of fault modes while still enabling adaptation to accommodate unanticipated faults.

Reconfigurable path planning provides a means to generate flight paths based on the degraded capability of the aircraft. Attempting to recover the capability of the nominal system after the occurrence of a fault is not a realistic course of action. The reconfigurable path planning component generates adaptive flight paths based on the capability of the vehicle in real-time. The reconfigurable path planner implemented in Chapter 6 optimized the GTMax's longitudinal dynamics, but extension to lateral dynamics is clearly possible. Further extensions in reconfigurable path planning could include non-linear methods to generate flight paths. Row 4 in Table 4 quantifies the benefit of reconfigurable path planning component after the occurrence of a fault. Notice that the maximum vertical error is lower in Row 4 than in Row 1, where no fault was applied. This result supports the application of reconfigurable path planning on the baseline system.

Like previous hierarchical fault-tolerant control architectures, this one is expandable vertically. However, this architecture separates itself from its predecessors by providing a well defined interface with higher level algorithms. The mission assignment component (Figure 5) in the third tier of the hierarchy receives sequences of waypoints from higher tier algorithms. In exchange, the mission adaptation component conducts an online assessment and captures the capability of the aircraft in a simple model. That model is available to higher tiers in the hierarchy as a small set of exportable parameters. The DARPA HURT program is currently constructing an architecture for controlling multiple heterogeneous UAVs in an urban environment. In its early stages, the HURT program recognized that simple aircraft models are an essential ingredient for conducting high level mission planning tasks. By providing a simple aircraft model, the fault-tolerant control architecture developed in this research facilitates increasing the level of autonomy in unmanned aircraft.

The following list articulates the major contributions of this research:

- An integrated fault-tolerant architecture that incorporates fault detection and identification, active system restructuring, reconfigurable flight controllers, reconfigurable path planning, and mission adaptation to optimize the usefulness of the aircraft.

- An active system restructuring component that maximizes vehicle performance in the presence of a fault without degrading the performance of the nominal system.
- A suite of adaptive reconfigurable flight controllers with guaranteed stability properties that employ active control to augment the controllability of the degraded system.
- A reconfigurable path planning component that generates flight paths within the capability of the degraded system.
- An online system identification process designed to enhance path planning and aid higher level decision making processes.
- A mission adaptation component that imposes constraints on the closed loop performance expectations of the aircraft.

8.2 Application of the Fault-Tolerant Control Architecture to UAV Upset Recovery

The fault-tolerant control process is readily applicable to another area of recent UAV research, upset recovery. In a sense, upsets such as wind gusts and wake turbulence are external faults, and the fault-tolerant control architecture developed in this research possesses all the essential components to conduct successful upset recovery. The architecture must include an FDI component capable of detecting and identifying upsets using air data and other sensor inputs. Part of this identification process includes estimating the wind velocity and direction. Upsets also differ from faults in that they are usually temporary in nature so the FDI component needs a capability to identify when an upset has terminated. In the presence of an upset, a typical response involves abandoning precise position control in favor of attitude control. The active system restructuring component would implement this change by assigning reconfigurable flight controllers to assume outer loop control functions in the baseline controller. The reconfigurable path planner could generate flight paths that minimize power changes while the mission adaptation component revises waypoints to reflect the degraded capability of the aircraft. The procedures for successful upset recovery would involve every component in the architecture.

8.3 Integration with Higher Level Control Algorithms

In the current hierarchical control architecture, human operators interface with the third tier by providing sequences of waypoints for the vehicle. The second tier generates a flight path from the upcoming waypoints, and the lowest tier generates actuator inputs for the UAV. Extending this progression from actuator inputs to sequences of waypoints upward will increase the autonomy of the unmanned system and move the human interface to a higher tier in the hierarchy. Following the current progression, the fourth tier would generate a sequence of waypoints from a human operator input. This level of abstraction will consider factors such as terrain, weather, the location of enemy positions, fuel consumption, and the health of the aircraft to generate a viable sequence of waypoints.

8.3.1 Post-fault mission planning

The first chapter of this work discussed the typical procedures which human pilots apply in event of in-flight emergencies. The fifth step, 'Decide on a course of action' was not addressed in the three-tier hierarchical architecture. A four-tier fault-tolerant control architecture should address this step. Evaluation of the problem will consider the capability of the degraded aircraft, the prognosis of the aircraft given its current fault mode, and the value of the airframe compared to the value of the mission. The fourth tier component should strive to maximize the accomplishment of assigned mission tasks without exceeding an allowable risk threshold. Simple risk analysis could evaluate the three options mentioned in Chapter 1: Land as soon as possible, Land as soon as practicable, and Continue the mission. More complex analysis would enable the aircraft to continue its mission in some degraded fashion with an acceptable risk.

Multi-ship UAV operations compound the difficulty of post-fault mission planning. The occurrence of a fault in a single aircraft can demand a new course of action for an entire team of unmanned vehicles. As research moves towards multi-ship UAV operations, contingency management will dictate the development of high level mission planners.

8.3.2 Fault-preventive mission planning for risk mitigation

Fault-preventive mission planning poses to apply risk assessment and mitigation to the UAV mission planning process. Simple applications of fault-prevention such as avoiding icing conditions or adverse winds may reside at the fourth tier of the control hierarchy. Other more complex methods would require a higher-level of autonomy which enables considering the disposition of enemy forces, collision avoidance, the proximity of humans, and the availability of suitable emergency landing areas among other factors. Providing UAVs with the autonomy to select suitable landing areas is an active area of research. Fault-preventive mission planning will account for contingency management during the planning and/or re-planning phase.

APPENDIX A

ALGORITHMS FOR SYSTEM IDENTIFICATION

Linear regression and Kalman filtering are the two most prevalent methods to attack the system identification problem. For the following discussion on regression, the subscript "*reg*" is employed to avoid ambiguity with previously assigned variables. Equation 4 is restated as:

$$y_{reg}(n) = \theta_{reg}(n)^T \phi_{reg}(n) + v(n) \tag{34}$$

where $\phi_{reg}(n)$ is the regressor vector $[x^T, u^T]^T$, $\theta_{reg}^T(n)$ is the current estimate of a row of $[A_f, B_f]$, and $v(n)$ is a residual error due to sensor noise and modeling error. In this case, $y_{reg}(n)$ is an element of $\dot{x}_f$. When certain values of the A_f and B_f matrices are known *à priori* with certainty, they can be moved to the left side of the equation thus reducing the length of $\theta_{reg}(n)$ and $\phi_{reg}(n)$ and simplifying the estimation process. The least square algorithm drives $\theta_{reg}(n)$ towards its true value, θ^* by minimizing the squared error:

$$J(\theta_{reg}) = \frac{1}{2} \sum_{k=n-N+1}^{n} [r(k)^T r(k)] \tag{35}$$

$$r(k) = y_{reg}(k) - \theta_{reg}(n)^T \phi_{reg}(k) \tag{36}$$

where N is the length of the observation window. The sequential least squares solution to Equation 34 is:

$$\theta^* = (H^T H)^{-1} H^T \bar{y}_{reg} \tag{37}$$

where $H = [\phi_{reg}(n-N+1), \phi_{reg}(n-N+2), ..., \phi_{reg}(n)]^T$ and
$\bar{y}_{reg} = [y_{reg}(n-N+1), y_{reg}(n-N+2), ..., y_{reg}(n)]^T$. It is well known that this solution can also be achieved recursively thereby avoiding matrix inversion [45].

The signals commonly encountered in reconfigurable flight "pose significant problems" for system identification [79]. Flight control architectures require fast and accurate parameter identification, and the signals frequently lack information content. To overcome these shortcomings, various improvements to the least squares algorithm have been proposed. One method employs a singular value decomposition method to bypass problems that occur when $(H^T H)$ is nearly singular [16, 68]. This situation, which is common in reconfigurable flight control, occurs when one or more of the regressors are linearly dependent. The singular value decomposition of H is used to discard information that is not useful. The Modified Sequential Least Squares (MSLS) algorithm seeks to minimize an augmented cost function [77]:

$$J(\theta_{reg}) = \frac{1}{2} \sum_{k=n-N+1}^{n} [r(k)^T r(k) + p(k)^T W_0 p(k) + q(k)^T W_1 q(k)] \tag{38}$$

$$p(k) = \theta_{reg}(k) - \theta_{reg}(k-1)$$
$$q(k) = \theta_{reg}(k) - \hat{\theta}$$

where $\hat{\theta}$ is an *à priori* estimate of θ^*. W_0 and W_1 are weighting matrices. The augmented cost function includes two additional terms. The first additional term restricts the movement of the estimate, θ_{reg}, in the temporal dimension and the second in the spatial dimension. Seemingly, MSLS would slow the reaction time and degrade the accuracy of the parameter identification, but results actually show that MSLS out-performs least squares even when $\theta^* \neq \hat{\theta}$ [79]. The second additional term performs a task called *regularization*. Several sources in the literature employ regularization as a means to incorporate *à priori* information in estimates [16, 68, 76, 79]. In the simplest methods, $\hat{\theta}$ is set to the best estimate of the unimpaired θ^*. Other algorithms incorporate additional *à priori* information that is drawn from the linearization of non-linear kinematics [79] or from a polynomial neural network that is trained offline [77]. Research provides a comprehensive linearization for the flight dynamics of a small unmanned rotorcraft [9, 30]. Linear regression with intercepts allows the estimation to account for unknown bias terms such an aircraft trim condition [51]. When the methods described thus far fail to extract sufficient information

from the system, the designer can intentionally stimulate the control inputs to inject information. Stimulating the null space of the control matrix B does not affect the response of the vehicle [20].

All the identification methods discussed thus far involve minimization of an error function in the time domain. Fourier transform regression (FTR) conducts identification in the frequency domain [59, 70]. The algorithm employs the standard least squares solution to extract estimates from discrete Fourier transforms of the regressor signals. A recursive formulation for the Fourier transform reduces the computation involved [70]. FTR enables the control designer to select which frequencies to include in the regression. In fact, the algorithm is quite sensitive to the designer's choice of frequencies. For determination of flight stability and control derivatives, the high frequency components of the regressors are discarded. Injection of constant frequency disturbances to the control vector can be used to stimulate the frequency domain estimation process. In this case, the injection frequencies can be weighted in the estimation process [50, 70]. In support of NASA's IFCS program, FTRs had mixed results. During actual flight test, the algorithm was very successful in the identification of certain derivatives and unsuccessful on others.

Kalman filtering is a popular alternative to linear regression. Multiple sources employ Kalman filters to estimate portions of the B_f matrix [57, 87]. The process extends the classical Kalman filter that estimates the aircraft state. Adaptive parameter smoothing uses the parameter error variance to adaptively tune the filter's forgetting factor. The algorithm improves the parameter estimation and precludes bursting [57]. Kalman filters are easier to implement than regression algorithms, and state estimation Kalman filters often already exist with the control architecture. A variety of modern algorithms such as fuzzy logic, neural networks, and particle filters are also well-suited for the state estimation problem. Gutièrrez employed fuzzy neural networks to update each of the linear models in his AMTC architecture. A recursive least squares method that he developed was used to train the fuzzy neural networks [32].

APPENDIX B

DISCUSSION OF TAIL ROTOR MALFUNCTIONS

This research did not address tail rotor malfunctions although restructuring to accommodate tail rotor malfunctions is possible. Several manned aircraft include emergency procedures for stuck left pedal (high power) and stuck right pedal (low power) faults. Incidentally, these malfunctions occur more commonly than other actuator malfunctions because the linkage through the tail boom is longer and more prone to damage. Two characteristics specific to the tail rotor malfunction make reconfigurable flight control challenging. First, most procedures that involve tail rotor malfunctions terminate to a running/rolling landing allowing airspeed to oppose the errand yaw condition. Unmanned rotorcraft have not reached a level of autonomy where they can conduct running landings especially in the presence of a fault. Second, using RPM control to compensate for a tail rotor malfunction creates a non-minimum phase system. Increasing the main rotor RPM, ($\dot{\Omega} > 0$) causes a right yawing moment on a conventional helicopter whereas the actual increase in tail rotor RPM ($\Delta\Omega > 0$) creates a left yawing moment. Despite these challenges a reconfigurable controller that enables aircraft recovery and continued forward flight in the presence of a tail rotor fault is certainly possible.

APPENDIX C

SELECTION OF RECONFIGURABLE FLIGHT CONTROLLER MATRICES P AND Q

Appendix B of [37] describes a method for selecting the matrices P and Q for a second order adaptive neural network flight controller. This appendix motivates a selection for $P \in \Re^{3\times3}$ and $Q \in \Re^{3\times3}$ for the reconfigurable flight controller described in Chapter 4. When the dynamics of the system are linearized about the notional equilibrium point where the model in Equation 10 matches the actual model, and all the neural network weights are zero, the following system results:

$$\ddot{e} = A\dot{e} - kB\Gamma_W B^T P e \tag{39}$$

where A is given in Equation 14; $B^T = [0, 0, 1]$; and $\Gamma_W = I$ is a learning rate for the neural network. $k = (\frac{1}{4}n_2 + b)$ is a scalar constant. n_2 is the number of hidden layer neurons in the neural network; b value of the bias term in the hidden layer. P has the following form:

$$P = \begin{bmatrix} p_1 & p_{12} & p_{31} \\ p_{12} & p_2 & p_{32} \\ p_{31} & p_{32} & p_3 \end{bmatrix}. \tag{40}$$

Therefore, the third row of Equation 39 expands to:

$$\dddot{e}_1 + K_a\ddot{e}_1 + (K_v + kp_3)\ddot{e}_1 + (K_p + kp_{32})\dot{e}_1 + kp_{31}e_1 = 0, \tag{41}$$

where $e_1 = z - z_{rm}$ which has the form,

$$(s^2 + 2\omega_1\zeta_1 s + \omega_1^2)(s^2 + 2\omega_2\zeta_2 s + \omega_2^2) = 0 \tag{42}$$

or equivalently,

$$s^4 + (2\omega_1\zeta_1 + 2\omega_2\zeta_2)s^3 + (\omega_1^2 + 4\omega_1\omega_2\zeta_1\zeta_2 + \omega_2^2)s^2 + (2\omega_1\zeta_1\omega_2^2 + 2\omega_2\zeta_2\omega_1^2)s + \omega_1^2\omega_2^2 = 0. \tag{43}$$

Equation 41 leaves the designer three degrees of freedom (p_{31}, p_{32}, p_3) to place four parameters of the fourth order linear system, (ω_1, ω_2, ζ_1, ζ_2). Choosing to set $\zeta_1 = \zeta_2 = 1$ and $\omega_1 = \omega_n$, where ω_n is the bandwidth of the controller, leads to the following:

$$\begin{aligned} p_{31} &= \frac{1}{k}\omega_1^2\omega_2^2 \\ p_{32} &= \frac{2}{k}\omega_1\omega_2^2 \\ p_3 &= \frac{1}{k}\omega_2^2 \end{aligned} \tag{44}$$

where $K_a = 2\omega_1\zeta_1 + 2\omega_2\zeta_2$, implies $\omega_2 = \frac{1}{2\tau}$. Enforcing that $Q = -A^TP - PA$ is diagonal places the remaining terms in P,

$$\begin{aligned} p_1 &= \frac{1}{k}(\omega_1^4\omega_2^2 + 8\omega_1^3\omega_2^3) \\ p_{12} &= \frac{1}{k}(4\omega_2^3\omega_1^2 + 2\omega_1^3\omega_2^2) \\ p_2 &= \frac{1}{k}(8\omega_1\omega_2^3 + 4\omega_1^2\omega_2^2). \end{aligned}$$

The following Q is obtained:

$$Q = \frac{4}{k}\begin{bmatrix} \omega_1^4\omega_2^3 & 0 & 0 \\ 0 & 2\omega_1^2\omega_2^3 & 0 \\ 0 & 0 & \omega_2^3 \end{bmatrix} = \frac{(K_a - 2\omega_1)^3}{2k}\begin{bmatrix} \omega_1^4 & 0 & 0 \\ 0 & 2\omega_1^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{45}$$

which is positive definite because $\omega_2 = \frac{1}{2\tau} > 0$. Note these relatively simple expressions for P and Q only hold if ω_1 precisely equals ω_n. Nonetheless, assuming that the expression for Q is constant over variations in ω_1 results in a simple optimization problem. Setting $\omega_1 = 1$, maximizes the minimum eigenvalue of Q over a wide range of control parameters ω_n and τ, namely if $K_a > 3.5$.

Based on this result, the simple expressions for p_{31}, p_{32}, and p_3 in Equation 44 and $\omega_1 = 1$ were used in the GTMax RPM controller. Doing so dictates new values for p_1, p_{12}, and p_2; a diagonal Q matrix is still achievable. Figure 34 depicts the fluctuation of the eigenvalues of Q over ω_1 using $\omega_n = 1.25$ and $K_a = 5$, the settings on the GTMax. The actual maximum of $\lambda_{min}(Q)$ occurs near $\omega_1 = 1$. $\lambda_{min}(Q)$ is nearly doubled by using $\omega_1 = 1$, not $\omega_1 = \omega_n$. Also, note that a poor choice of ω_1 can render a Q that is not positive definite, a case which does not occur with $\omega_1 = \omega_n$.

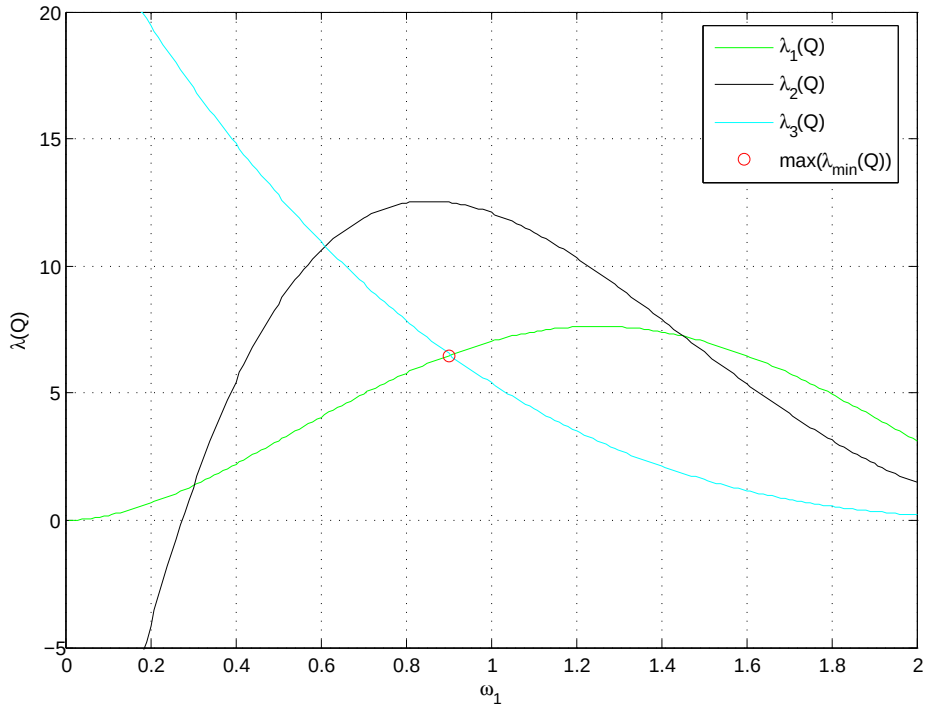

Figure 34. Fluctuation of the eigenvalues of Q as a function of ω_1 with $K_a = 5$.

APPENDIX D

DEVELOPMENT OF AN ALTERNATE RECONFIGURABLE FLIGHT CONTROLLER

An alternate RPM controller updates an adaptive neural network to minimize model error, not tracking error. The controller depends on the assumption that a neural network converges using standard back propagation. Unlike the controller developed on Chapter 4 it does not have guaranteed stability properties. In the event the neural network fails to converge, a PID controller is activated. Through several flight tests, the control strategy performed well and the neural network did not diverge.

The baseline throttle loop controller achieves approximate adherence to the single pole linear system:

$$\dot{\Omega} = \frac{1}{\tau}(\Omega_{com} - \Omega). \tag{46}$$

The control Ω_{com} to achieve a desired rotor angular rate, Ω_{des} is thus:

$$\Omega_{com} = \tau\dot{\Omega}_{des} + \Omega_{des}. \tag{47}$$

A feedback linearization type controller is used to control the vertical thrust of the degraded system. Assuming the plant has affine dynamics:

$$\dot{w} = f + g\Omega \tag{48}$$

where w is the translational velocity of the vehicle in the z direction. The following control is applied to the system:

$$\Omega_{des} = \frac{1}{g'}[a_{rm_z} - f' - K_d(w - w_{rm}) - K_p(z - z_{rm})]. \tag{49}$$

where a_{rm_z}, w_{rm}, and z_{rm} indicate the desired vertical dynamics of the vehicle. A second order filter is normally applied to these dynamics prior to inclusion in the control law above.

f' and g' are estimates of the actual f and g such that

$$\dot{w}' = f' + g'\Omega. \tag{50}$$

f' is estimated by a linear combination of the state and control vectors plus the output of the adaptive neural network . The input layer of the network consists of the vehicle state and control vectors and the linear estimate. Back propagation updates the single hidden layer network to minimize the model error, ϵ. g' is approximated simply as a constant. Scheduling g' with horizontal velocity has also proved to be beneficial in simulation.

$$\epsilon = \dot{w} - \dot{w}' = \Delta f + \Delta g\Omega \tag{51}$$

Combining Equation 47 and Equation 49 and letting $e_1 = z - z_{rm}$, one obtains the following:

$$\begin{aligned}\Omega_{com} = \frac{1}{g'}[\tau\ddot{w}_{com} + \dot{w}_{com} - \tau\dot{f}' - f' - \tau K_d\ddot{e}_1 \\ -(\tau K_p + K_d)\dot{e}_1 - K_p e_1].\end{aligned} \tag{52}$$

The error dynamics of the system reduce to:

$$\dddot{e}_1 = -(K_d + \frac{1}{\tau})\ddot{e}_1 - (\frac{K_d}{\tau} + K_p)\dot{e}_1 - \frac{K_p}{\tau}e_1 + \epsilon/\tau + \dot{\epsilon}. \tag{53}$$

Assuming convergence of the network, ϵ and $\dot{\epsilon}$ are negligible. The characteristic equation takes the following form:

$$(s + \frac{1}{\tau})(s^2 + 2\omega_n\zeta s + \omega_n^2) = 0. \tag{54}$$

K_p and K_d are set as follows to dictate the performance of the second order system:

$$K_p = \omega_n^2 \tag{55}$$

$$K_d = 2\omega_n\zeta. \tag{56}$$

REFERENCES

[1] "Unmanned aerial vehicles roadmap 2002-2027," tech. rep., Office of the Secretary of Defense, Washington, DC 20301, 2002.

[2] "Unmanned aircraft systems roadmap 2005-2030," tech. rep., Office of the Secretary of Defense, Washington, DC 20301, 2005.

[3] Berger, T. K., Tigner, B., and Glusman, S., "Vtol uav applications and operations: The boeing maverick," in *Proceedings of the AHS International Specialists' Meeting on Unmanned Rotorcraft*, (Chandler, AZ), January 18-20, 2005.

[4] Black, S. E., Keller, K., Biswas, G., and Davis, J., "Diagnostic/prognostic modeling and reconfigurable control," in *Proceedings of the IEEE AUTOTESTCON*, (San Antonio, TX), pp. 344–350, September 2004.

[5] Bodson, M., "Evaluation of optimization methods for control allocation," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-2001-4223, (Montreal, Canada), August 6-9, 2001.

[6] Bodson, M., "Self-designing control systems for piloted and uninhabited aerial vehicles," Tech. Rep. AD-A387960, AFRL-SR-BL-TR-01-0176, Atlanta, GA, 2001.

[7] Bodson, M. and Groszkiewicz, J. E., "Multivariable adaptive algorithms for reconfigurable flight control," *IEEE Transations on Control Systems Technology*, vol. 5, pp. 217–229, March 1997.

[8] Bodson, M. and Pohlchuck, W. A., "Command limiting in adaptive control," *Journal of Guidance, Control, and Dynamics*, vol. 21, pp. 639–646, July-August 1998.

[9] Bogdanov, A., Carlsson, M., Harvey, G., Hunt, J., Kieburtz, D., van der Merwe, R., and Wan, E., "State-dependent riccati equation control of a small unmanned helicopter," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-2003-5672, (Austin, TX), August 11-14, 2003.

[10] Bošković, J. D., Li, S.-M., and Mehra, R. K., "On-line failure detection and identification (FDI) and adaptive reconfigurable control (ARC) in aerospace applications," in *Proceedings of the American Control Conference*, (Arlington, VA), June 25-27, 2001.

[11] Bošković, J. D., Li, S.-M., and Mehra, R. K., "Robust adaptive variable structure control of spacecraft under control input saturation," *Journal of Guidance, Control, and Dynamics*, vol. 24, pp. 14–22, January-February 2001.

[12] Bošković, J. D. and Mehra, R. K., "A multiple model-based reconfigurable flight control system design," in *Proceedings of the 37th IEEE Conference on Deciscion and Control*, (Tampa, FL), December 1998.

[13] Bošković, J. D., Prasanth, R., and Mehra, R. K., "A multi-layer control architecture for unmanned aerial vehicles," in *Proceedings of the American Control Conference*, (Anchorage, AK), May 8-10, 2002.

[14] Brinker, J. S. and Wise, K. A., "Reconfigurable flight control for a tailless advanced fighter aircraft," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-1998-4107, (Boston, MA), pp. 75–87, August 10-12, 1998.

[15] Brinker, J. S. and Wise, K. A., "Flight testing of a reconfigurable flight control law on the X-36 tailless fighter aircraft," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-2000-3941, (Denver, CO), pp. 1–11, August 14-17, 2000.

[16] Buffington, J. and Chandler, P., "Integration of on-line system identification and optimization-based control allocation," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-1998-4487, (Boston, MA), pp. 1746–1756, August 10-12, 1998.

[17] Calise, A. J., Lee, S., and Sharma, M., "Development of a reconfigurable flight control law for tailless aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 5, pp. 896–902, 2001.

[18] Celi, R., Huang, C., and Shih, I. C., "Reconfigurable flight control systems for a tandem rotor helicopter," in *Proceedings of the American Helictoper Society 52nd Annual Forum*, (Washington, DC), June 4-6, 1996.

[19] Clements, N. S., *Fault Tolerant Control of Complex Dynamical Systems*. PhD thesis, Georgia Institute of Technology, School of Electrical and Computer Engineering, Atlanta, GA 30332, April 2003.

[20] Doman, D. B. and Ngo, A. D., "Dynamic inversion-based adaptive/reconfigurable control of the X-33 on ascent," *Journal of Guidance, Control, and Dynamics*, vol. 25, pp. 275–284, March 2002.

[21] Drozeski, G., Saha, B., and Vachtsevanos, G., "Fault detection and reconfigurable control architecture for unmanned aerial vehicles," in *Proceedings of the IEEE Aerospace Conference*, (Big Sky, MT), March 5-12, 2005.

[22] Drozeski, G., Saha, B., and Vachtsevanos, G., "Fault tolerant architecture for an unmanned rotorcraft," in *Proceedings of the AHS International Specialists' Meeting on Unmanned Rotorcraft*, (Chandler, AZ), January 18-20, 2005.

[23] Drozeski, G. and Vachtsevanos, G., "Fault-tolerant architecture with reconfigurable path planning applied to an unmanned rotorcraft," in *Proceedings of the American Helicopter Society 61st Annual Forum*, (Grapeview, TX), June 1-3, 2005.

[24] Drozeski, G., Yavrucuk, I., Johnson, E., Prasad, J., Schrage, D., and Vachtsevanos, G., "Application of software enabled control technologies to a full-scale unmanned helicopter," in *Proceedings of the AIAA Atmospheric Flight Mechanics Conference and Exhibit*, no. AIAA-2002-6423, (San Francisco, CA), August 15-18, 2005.

[25] Elgersma, M. and Glavaski, S., "Reconfigurable control for active management of aircraft system failures," in *Proceedings of the American Control Conference*, (Arlington, VA), June 25-27, 2001.

[26] Enns, R., "Helicopter reconfigurable flight control with actuator geometry optimization," in *Proceedings of the American Helicopter Society Flight Controls and Crew System Design Specialists' Meeting*, (Philadelphia, PA, USA), October 9-11, 2002.

[27] Enns, R. and Si, J., "Helicopter flight control design using a learning control approach," in *Proceedings of the IEEE Conference on Decision and Control*, (Sydney, Australia), December 2000.

[28] Enns, R. J., *Neural Dynamic Programming Applied to Rotorcraft Flight Control and Reconfiguration*. PhD thesis, Arizona State University, Department of Electrical Engineering, Tempe, Az, December 2001.

[29] Gao, Z. and Antsaklis, P., "On the stability of the pseudo-inverse method for reconfigurable control systems," in *Proceedings of the IEEE National Aerospace and Electronic Conference (NAECON)*, (Dayton, OH), May 22-26, 1989.

[30] Gavrilets, V., Mettler, B., and Feron, E., "Nonlinear model for a small-size acrobatic helicopter," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Montreal, Canada), August 6-9, 2001.

[31] Gili, P. A. and Battipede, M., "Adaptive neurocontroller for a nonlinear combat aircraft model," *Journal of Guidance, Control, and Dynamics*, vol. 24, pp. 910–917, September-October 2001.

[32] Gutièrrez-Zea, L., *An Architecture for Adaptive Mode Transition Control of Unmanned Aerial Vehicles*. PhD thesis, Georgia Institute of Technology, School of Electrical and Computer Engineering, Atlanta, GA 30332, 2004.

[33] Heiges, M. W., "Reconfigurable control for rotorcraft," Tech. Rep. Project Number A9583, Georgia Tech Research Institute, Georgia Institute of Technology, Atlanta, GA, 1995.

[34] Hess, R., Siwakosit, W., and Chung, J., "Accommodating a class of actuator failures in flight control systems," *Journal of Guidance, Control, and Dynamics*, vol. 23, pp. 412–419, May-June 2000.

[35] Hess, R. and S.R.Wells, "Sliding mode control applied to reconfigurable flight control design," *Journal of Guidance, Control, and Dynamics*, vol. 26, pp. 452–462, May-June 2003.

[36] Johnson, E., Schrage, D., and Vachtsevanos, G., "Rotary wing final experiments for the software enabled control program," in *Proceedings of the American Helicopter Society 61st Annual Forum*, (Grapeview, TX), June 1-3, 2005.

[37] Johnson, E. N., *Limited Authority Adaptive Flight Control*. PhD thesis, Georgia Institute of Technology, School of Aerospace Engineering, Atlanta, GA 30332, November 2000.

[38] Johnson, E. N., Calise, A. J., Mease, K. D., Corban, J. E., and Curry, M. D., "Adaptive guidance and control for hypersonic vehicles," *Journal of Guidance, Control, and Dynamics*, (accepted).

[39] Johnson, E. N. and Kannan, S. K., "Adaptive flight control for an autonomous unmanned helicopter," in *Proceedings of the AIAA Guidance Navigation and Control Conference and Exhibit*, no. AIAA-2002-4439, (Monterey, CA), August 2002.

[40] Johnson, E. N. and Kannan, S. K., "Adaptive trajectory control for autonomous helicopters," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 3, pp. 524–538, 2005.

[41] Jorgensen, C. C., "Direct adaptive aircraft control using dynamic cell structure neural networks," Tech. Rep. NASA/TM-1997-112198, National Aeronautics and Space Administration, Ames Research Center, Moffett Field, CA, October 2003.

[42] Kim, H. S. and Kim, Y., "Partial eigenstructure assignment algorithm in flight control system design," *IEEE Transations on Aerospace and Electronic Systems*, vol. 35, pp. 1403–1409, October 1999.

[43] Kim, N., Calise, A., Corban, J., and Prasad, J., "Adaptive output feedback for altitude control of an unmanned helicopter using rotor RPM," in *Proceedings of the AIAA Guidance Navigation and Control Conference and Exhibit*, no. AIAA-2004-5323, (Providence, RI), August 16-19, 2004.

[44] Krishnakumar, K. and Gundy-Burlet, K., "Intelligent control approaches for aircraft applications," Tech. Rep. N20020054263, National Aeronautics and Space Administration, Ames Research Center, Moffett Field, CA, 2001.

[45] Lee, S. and Choi, J., "Indirect adaptive control using eigenstructure assignment," in *Proceedings of the 40th Annual Conference of the Society of Instrument and Control Engineers*, (Nagoya, Japan), 2001.

[46] Liao, F., Wang, J. L., Poh, E. K., and Li, D., "Reliable automatic landing control against actuator stuck faults," in *Proceedings of the AIAA Guidance Navigation and Control Conference and Exhibit*, no. AIAA-2004-5233, (Providence, RI), August 16-19, 2004.

[47] Maki, M., Jiang, J., and Hagino, K., "A stability guaranteed active fault tolerant control system against actuator failures," in *Proceedings of the Conference on Decision and Control*, (Orlando, FL), December 2001.

[48] McFarland, M. B. and Calise, A. J., "Multilayer neural networks and adaptive nonlinear control of agile anti-air missiles," in *Proceedings of the AIAA Guidance Navigation and Control Conference*, no. AIAA-1997-3540, (New Orleans, LA), August 11-13, 1997.

[49] Miller, R. H. and Larsen, M. L., "Optimal fault detection and isolation filters for flight vehicle performance monitoring," in *Proceedings of the IEEE Aerospace Conference*, vol. 7, pp. 3197–3203, March 8-15, 2003.

[50] Moes, T. R., Smith, M. S., and Morelli, E. A., "Flight investigation of prescribed simultaneous independent surface excitations for real-time parameter estimation," Tech. Rep. NASA/TM-2003-212029, National Aeronautics and Space Administration, Dryden Flight Research Center, Edwards, CA 93523-0273, October 2003.

[51] Nelson, E. B. and Pachter, M., "Linear regression with intercept," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-2004-4757, (Providence, RI), August 16-19, 2004.

[52] Oppenheimer, M. W., Doman, D. B., and Bolander, M. A., "A method for estimating control failure effects for aerodynamic vehicle trajectory retargeting," Tech. Rep. AFRL-VA-WP-TP-2004-302, Air Force Research Lab, Air Vehicles Directorate, Wright-Patterson Air Force Base, CA 93523-0273, February 2004.

[53] Osder, S., "Practical view of redundancy management application and theory," *Journal of Guidance, Control, and Dynamics*, vol. 22, pp. 12–21, January-Febraury 1999.

[54] Osder, S. S., "Reconfigurable helicopter flight control system." U. S. Patent 5,678,786, Oct. 1997.

[55] Pachter, M., Chandler, P. R., and Mears, M., "Reconfigurable tracking control with saturation," *Journal of Guidance, Control, and Dynamics*, vol. 18, no. 5, 1995.

[56] Pachter, M., Chandler, P. R., and Smith, L., "Maneuvering flight control," *Journal of Guidance, Control, and Dynamics*, vol. 21, pp. 368–374, May-June 1998.

[57] Pachter, M. and Huang, Y.-S., "Fault tolerant flight control," *Journal of Guidance, Control, and Dynamics*, vol. 26, pp. 151–160, January-February 2003.

[58] Page, A. and Steinberg, M. L., "A closed-loop comparison of control allocation methods," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-2000-4538, (Denver, CO), August 14-17, 2000.

[59] Perhinschi, M. G., Lando, M., Massoti, L., Campa, G., Napolitano, M. R., and Fravolini, M. L., "On-line parameter estimation issues for the NASA IFCS F-15 fault tolerant systems," in *Proceedings of the American Control Conference*, (Anchorage, AK), May 8-10, 2002.

[60] Petersen, J. A. M. and Bodson, M., "Interior point algorithms for control allocation," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-2001-5566, (Austin, TX), August 11-14, 2003.

[61] Proctor, A. and Johnson, E. N., "Latency compensation in an adaptive flight controller," in *Proceedings of the AIAA Guidance Navigation and Control Conference and Exhibit*, no. AIAA-2003-5413, (Austin, TX), August 11-14, 2003.

[62] Reimann, J. and Vachtsevanos, G., "Computational methods in pursuit-evasion problems," in *Proceedings of the International Conference on Computational Methods in Science and Engineering*, (Loutraki, Greece), October 21-26, 2005.

[63] Sastry, S. and Bodson, M., *Adaptive Control: Stability, Convergence and Robustness*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[64] Schierman, J. D., Hull, J. R., Gandhi, N., and Ward, D. G., "Flight test results of an adaptive guidance system for reusable launch vehicles," in *Proceedings of the AIAA Guidance Navigation and Control Conference and Exhibit*, no. AIAA-2004-4771, (Providence, RI), August 16-19, 2004.

[65] Schrage, D. P. and Vachtsevanos, G., "Software enabled control for intelligent UAVs," in *Proceedings of the 1999 IEEE International Conference on Control Applications*, 1999.

[66] Shoureshi, R., Wheeler, M., Bell, M., Alves, G., and Maguire, D., "On implementation of active control systems," in *Proceedings of the American Control Conference*, (Seattle, WA), June 1995.

[67] Shtessel, Y., Buffington, J., Pachter, M., Chandler, P., and Banda, S., "Reconfigurable flight control on sliding modes addressing actuator deflection and deflection rate saturation," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, vol. 1, (Boston, MA), pp. 1466–1480, August 10-12, 1998.

[68] Smith, C. L., Chandler, P. R., and Pachter, M., "Regularization techniques for real-time identification of aircraft parameters," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, vol. 1, (New Orleans, LA), pp. 1466–1480, August 11-13, 1997.

[69] Smith, D. J., *Reliability, Maintainability, and Risk: Practical Methods for Engineers*. Boston, MA: Butterworth-Heinemann, sixth ed., 2001.

[70] Smith, M. S., Moes, T. R., and Morelli, E. A., "Real-time stability and control derivative extraction from F-15 flight data," Tech. Rep. NASA/TM-2003-212027, National Aeronautics and Space Administration, Dryden Flight Research Center, Edwards, CA 93523-0273, September 2003.

[71] Sobel, K. and Shapiro, E., "Eigenstructure assignment for design of multimode flight control systems," *IEEE Control Systems Magazine*, vol. 5, pp. 9–15, May 1985.

[72] Vachtsevanos, G., Tang, L., Drozeski, G., and Gutièrrez, L., "Intelligent control of unmanned aerial vehicles for improved autonomy and reliability," in *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*, (Lisbon, Portugal), July 5-7, 2004.

[73] Vachtsevanos, G., Tang, L., Drozeski, G., and Gutièrrez, L., "From mission planning to flight control of unmanned aerial vehicles: Strategies and implementation tools," *Annual Reviews in Control*, vol. 29, pp. 101–115, 2005.

[74] Valenti, M., Mettler, B., Schouwenaars, T., Feron, E., and Paduano, J., "Trajectory reconfiguration for an unmanned aircraft," in *Proceedings of the AIAA Guidance Navigation and Control Conference and Exhibit*, no. AIAA-2002-4674, (Monterey, CA), August 2002.

[75] Wang, W., Jaw, L., and Wang, J., "Petri net based modeling approach for fault affected UAV subsystems reconfiguration," in *Proceedings of the AIAA Intelligent Systems Technical Conference*, no. AIAA-2004-6355, (Chicago, IL), September 20-22, 2004.

[76] Ward, D. G. and Barron, R. L., "A self-designing receding horizon optimal flight controller," in *Proceedings of the American Control Conference*, (Seattle, WA), June 1995.

[77] Ward, D. G., Monaco, J. F., and Schierman, J. D., "Reconfigurable control for VTOL UAV shipboard landing," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, vol. 1, (Portland, OR), pp. 499–509, August 9-11, 1999.

[78] Ward, D. G., Sharma, M., Richard, N. D., DeLuca, J., and Mears, M., "Intelligent control of unmanned air vehicles: Program summary and representative results," in *Proceedings of the 2nd AIAA "Unmanned Unlimited" Systems, Technologies, and Operations*, no. AIAA-2003-6641, (San Diego, CA), September 15-18, 2003.

[79] Ward, D. G. and Monaco, J. F., "Development and flight testing of a parameter identification algorithm for reconfigurable control," *Journal of Guidance, Control, and Dynamics*, vol. 21, pp. 948–956, November-December 1998.

[80] Wills, L., Kannan, S. K., Sander, S., Guler, M., Heck, B., Prasad, J., Schrage, D. P., and Vachtsevanos, G., "An open platform for reconfigurable control," vol. 21, June 2001.

[81] Yavrucuk, I. and Prasad, J. V. R., "Simulation of reconfigurable heli-UAVs using main rotor RPM control in failure modes," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, 1999.

[82] Yavrucuk, I., Prasad, J. V. R., and Calise, A. J., "Adaptive limit detection and avoidance for carefree maneuvering," in *Proceedings of the AIAA Atmospheric Flight Mechanics Conference and Exhibit*, no. AIAA-1997-3740, (Montreal, Canada), August 9-11, 2001.

[83] Yavrucuk, I., Prasad, J. V. R., and Hanagud, S. V., "Reconfigurable flight control using RPM control for heli-UAV's," in *Proceedings of the 25th European Rotorcraft Forum*, 1999.

[84] Zadeh, L. A. and Whalen, B. H., "On optimal control and linear programming," *IRE Transactions on Automatic Control*, vol. 7, pp. 45–46, July 1962.

[85] Zhang, Y. and Jiang, J., "Design of integrated fault detection, diagnosis and reconfigurable control systems," in *Proceedings of the 38th IEEE Conference on Decision and Control*, (Pheonix, AZ), December 1999.

[86] Zhang, Y. and Jiang, J., "An interacting multiple-model based fault detection, diagnosis and fault-tolerant control approach," in *Proceedings of the 38th IEEE Conference on Decision and Control*, vol. 4, (Pheonix, AZ), pp. 3593–3598, December 1999.

[87] Zhang, Y. and Jiang, J., "Design of proportional-integral reconfigurable control systems via eigenstruture assignment," in *Proceedings of the American Control Conference*, (Chicago, IL), June 2000.

[88] Zhang, Y. and Jiang, J., "Fault tolerant control system design with explicit consideration of performance degradation," *IEEE Transations on Aerospace and Electronic Systems*, vol. 39, pp. 838–848, July 2003.

RELATED PUBLICATIONS

DROZESKI, G., YAVRUCUK, I., JOHNSON, E., PRASAD, J.V.R., SCHRAGE,D., and VACHTSEVANOS, G.,"Application of software enabled control technologies to a full-scale unmanned helicopter", in *Proceedings of the AIAA Atmospheric Flight Mechanics Conference and Exhibit*, no. AIAA-2005-6234, (San Francisco, CA), August 15-18, 2005.

DROZESKI, G. and VACHTSEVANOS, G.,"A fault-tolerant architecture with reconfigurable path planning applied to an unmanned rotorcraft", in *Proceedings of the American Helicopter Society 61st Annual Forum*, (Grapevine, TX), June 1-3, 2005.

DROZESKI, G., SAHA, B., and VACHTSEVANOS, G.,"Fault tolerant architecture for an unmanned rotorcraft," in *Proceedings of the AHS International Specialists' Meeting on Unmanned Rotorcraft*, (Chandler, AZ), January 18-20, 2005.

DROZESKI,G., SAHA, B., and VACHTSEVANOS, G.,"Fault detection and reconfigurable control architecture for unmanned aerial vehicles," in *Proceedings of the IEEE Aerospace Conference*, (Big Sky, MT), March 5-12, 2005.

VACHTSEVANOS, G., TANG, L., DROZESKI, G., and GUTIERREZ, L.,"From mission planning to flight control of unmanned aerial vehicles: strategies and implementation tools", in *Annual Reviews in Control*, vol. 29 (2005), pp. 101-115.

VACHTSEVANOS, G., TANG, L., DROZESKI, G., and GUTIERREZ, L.,"Intelligent control of unmanned aerial vehicles for improved autonomy and reliability", in *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*, (Lisbon, Portugal), July 5-7, 2004.

VITA

Born on July 5, 1971, Graham Rolando Drozeski was raised in Columbia, South Carolina. He was awarded a Bachelor of Science in Electrical Engineering degree from the University of Notre Dame in 1993. Upon graduation, he accepted a commission in the United States Army. During nine years of service as an Army aviation officer, he piloted multiple military helicopters. From 2001-2002, he commanded a company of UH-60 Blackhawk helicopters at Fort Hood, Texas. He separated from the military in 2002 to pursue a graduate education. In 2003, he received a Master of Science degree from the School of Aerospace Engineering, Georgia Institute of Technology. He research interests include fault-tolerance, unmanned aerial vehicles, adaptive flight control, and intelligent control systems.